## ✅ TOP 200 PYTHON INTERVIEW PROGRAMS (QUESTION + CODE)

### 🔷 BASIC PROGRAMS (1–20)

**1. Print Hello World**

```
print("Hello World")
```

**2. Add two numbers**

```
a, b = 10, 20
print(a + b)
```

**3. Subtract two numbers**

```
print(20 - 10)
```

**4. Multiply two numbers**

```
print(5 * 4)
```

**5. Divide two numbers**

```
print(10 / 2)
```

**6. Swap two numbers**

```
a, b = 3, 4
a, b = b, a
```

**7. Check even or odd**

```
n = 7
print("Even" if n % 2 == 0 else "Odd")
```

**8. Check positive or negative**

```
n = -5
print("Positive" if n > 0 else "Negative")
```

**9. Find largest of two numbers**

```
print(max(10, 20))
```

**10. Find largest of three numbers**

```
print(max(1, 5, 3))
```

**11. Check leap year**

```
y = 2024
print(y % 4 == 0 and (y % 100 != 0 or y % 400 == 0))
```

**12. Find square**

```
print(5 ** 2)
```

**13. Find cube**

```
print(3 ** 3)
```

**14. Convert Celsius to Fahrenheit**

```
c = 25
print((c * 9/5) + 32)
```

**15. Convert km to miles**

```
km = 10
print(km * 0.621371)
```

**16. Simple interest**

```
p, r, t = 1000, 5, 2
print((p*r*t)/100)
```

**17. Area of circle**

```
import math
print(math.pi * 5 * 5)
```

**18. Area of rectangle**

```
print(5 * 10)
```

**19. ASCII value of character**

```
print(ord('A'))
```

**20. Character from ASCII**

```
print(chr(65))
```

## 🔷 NUMBER PROGRAMS (21–50)

**21. Factorial**

```
import math
print(math.factorial(5))
```

**22. Factorial (recursive)**

```
def fact(n): return 1 if n==0 else n*fact(n-1)
```

**23. Fibonacci series**

```
a,b=0,1
for _ in range(5):
    print(a,end=" ")
    a,b=b,a+b
```

**24. Prime number**

```
n=7
print(n>1 and all(n%i for i in range(2,int(n**0.5)+1)))
```

**25. Prime numbers in range**

```
for n in range(2,20):
    if all(n%i for i in range(2,int(n**0.5)+1)):
        print(n)
```

**26. Sum of digits**

```
print(sum(map(int,str(123))))
```

**27. Reverse a number**

```
print(int(str(123)[::-1]))
```

**28. Palindrome number**

```
n=121
print(str(n)==str(n)[::-1])
```

**29. Armstrong number**

```
n=153
print(n==sum(int(d)**len(str(n)) for d in str(n)))
```

**30. GCD**

```
import math
print(math.gcd(12,18))
```

**31. LCM**

```
a,b=12,18
print(a*b//math.gcd(a,b))
```

**32. Power without ****

```
res=1
for _ in range(3): res*=2
print(res)
```

**33. Count digits**

```
print(len(str(12345)))
```

**34. Sum of natural numbers**

```
n=10
print(n*(n+1)//2)
```

**35. Perfect number**

```
n=6
print(sum(i for i in range(1,n) if n%i==0)==n)
```

**36. Strong number**

```
import math
n=145
print(sum(math.factorial(int(d)) for d in str(n))==n)
```

**37. Binary to decimal**

```
print(int('1010',2))
```

**38. Decimal to binary**

```
print(bin(10)[2:])
```

**49. Kaprekar number**

**39. Octal to decimal**

```
print(int('12',8))
```

**40. Decimal to hexadecimal**

```
print(hex(255))
```

**41. Random number**

```
import random
print(random.randint(1,10))
```

**42. Count trailing zeros**

```
n=100
c=0
while n%10==0: c+=1; n//=10
print(c)
```

**43. Check automorphic**

```
n=25
print(str(n*n).endswith(str(n)))
```

**44. Harshad number**

```
n=18
print(n%sum(map(int,str(n)))==0)
```

**45. Neon number**

```
n=9
print(sum(map(int,str(n*n)))==n)
```

**46. Spy number**

```
n=1124
print(sum(map(int,str(n)))==eval("*".join(str(n))))
```

**47. Buzz number**

```
n=14
print(n%7==0 or str(n).endswith('7'))
```

**48. Duck number**

```
print('0' in str(102))
```

```
n=45
sq=str(n*n)
l=len(str(n))
print(int(sq[:-l] or 0)+int(sq[-l:])==n)
```

**50. Check ugly number**

```
n=6
for i in [2,3,5]:
    while n%i==0: n//=i
print(n==1)
```

---

## 🔷 STRING PROGRAMS (51–100)

(Condensed but complete)

**51. Reverse string**

```
print("python"[::-1])
```

**52. Palindrome string**

```
s="madam"
print(s==s[::-1])
```

**53. Count vowels**

```
print(sum(c in "aeiou" for c in "hello"))
```

**54. Count consonants**

```
print(sum(c.isalpha() and c not in "aeiou" for c in "hello"))
```

**55. Count characters**

```
from collections import Counter
print(Counter("hello"))
```

**56. Remove duplicates**

```
print("".join(dict.fromkeys("programming")))
```

**69. Remove vowels**

**57. Check anagram**

```
print(sorted("listen")==sorted("silent"))
```

**58. First non-repeating char**

```
s="aabbc"

print(next(c for c in s if s.count(c)==1))
```

**59. Replace spaces**

```
print("hello world".replace(" ","_"))
```

**60. Word count**

```
print(len("hello python world".split()))
```

**61. Capitalize words**

```
print("hello world".title())
```

**62. Toggle case**

```
print("PyThOn".swapcase())
```

**63. Check digit**

```
print("123".isdigit())
```

**64. Remove punctuation**

```
import string

print("hi!".translate(str.maketrans("","",string.punctuation)))
```

**65. Longest word**

```
s="I love python programming"

print(max(s.split(), key=len))
```

**66. Shortest word**

```
print(min(s.split(), key=len))
```

**67. String rotation**

```
s="abcd"

print("cdab" in s+s)
```

**68. Count substring**

```
print("banana".count("an"))
```

```
print("hello".translate(str.maketrans("","",'aeiou')))
```

**70. Check pangram**

```
import string

s="the quick brown fox jumps over lazy dog"

print(set(string.ascii_lowercase)<=set(s))
```

**71–100**

✅ PYTHON INTERVIEW PROGRAMS (71–200)
🔷 STRING PROGRAMS (71–100)
71. Sort characters in string

python
Copy code
```
s="python"
print("".join(sorted(s)))
```
72. Count uppercase letters

python
Copy code
```
print(sum(c.isupper() for c in "PyThOn"))
```
73. Count lowercase letters

python
Copy code
```
print(sum(c.islower() for c in "PyThOn"))
```
74. Remove digits from string

python
Copy code
```
s="abc123"
print("".join(c for c in s if not c.isdigit()))
```
75. Remove special characters

python
Copy code
```
import re
print(re.sub(r'[^a-zA-Z0-9 ]','',"hi@123"))
```
76. Check string contains only alphabets

python
Copy code
```
print("hello".isalpha())
```

77. Check string contains only numbers

```
python
Copy code
print("12345".isdigit())
```

78. Check string starts with substring

```
python
Copy code
print("python".startswith("py"))
```

79. Check string ends with substring

```
python
Copy code
print("python".endswith("on"))
```

80. Remove extra spaces

```
python
Copy code
print("hello   world".split())
```

81. Reverse words in string

```
python
Copy code
s="hello world"
print(" ".join(s.split()[::-1]))
```

82. Find duplicate characters

```
python
Copy code
s="programming"
print({c for c in s if s.count(c)>1})
```

83. String compression

```
python
Copy code
from itertools import groupby
s="aaabbc"
print("".join(c+str(len(list(g))) for c,g in groupby(s)))
```

84. Expand compressed string

```
python
Copy code
s="a3b2"
print("".join(s[i]*int(s[i+1]) for i in range(0,len(s),2)))
```

85. Check email format

```
python
Copy code
import re
print(bool(re.match(r'^(\d{1,3}\.){3}\d{1,3}$',"192.168.1.1"
```

87. Remove newline

```
python
Copy code
print("hello\n".strip())
```

88. Count punctuation

```
python
Copy code
import string
print(sum(c in string.punctuation for c in "hi!@#"))
```

89. Convert string to list

```
python
Copy code
print(list("python"))
```

90. Convert list to string

```
python
Copy code
print("".join(['p','y','t','h','o','n']))
```

91. Replace multiple spaces

```
python
Copy code
import re
print(re.sub(r'\s+',' ',"hello   world"))
```

92. Find ASCII values

```
python
Copy code
print([ord(c) for c in "abc"])
```

93. Remove leading zeros

```
python
Copy code
print(str(int("000123")))
```

94. Check substring

```
python
Copy code
print("py" in "python"))
```

import re
print(bool(re.match(r'^[\w.-]+@[\w.-]+\.\w+$',"a@gmail.com")))
86. Check IP address

95. String slicing

python
Copy code
print("python"[1:4])
96. String encoding

python
Copy code
print("hello".encode())
97. String decoding

python
Copy code
print(b'hello'.decode())
98. Check empty string

python
Copy code
s=""
print(len(s)==0)
99. Find index

python
Copy code
print("python".find("t"))
100. Replace character

python
Copy code
print("banana".replace("a","@"))
🔷 LIST PROGRAMS (101–140)
101. Sum of list

python
Copy code
print(sum([1,2,3]))
102. Largest element

python
Copy code
print(max([1,5,3]))
103. Smallest element

python
Copy code
print(min([1,5,3]))
104. Average of list

python
Copy code
lst=[1,2,3]
print(sum(lst)/len(lst))
105. Remove duplicates

python
Copy code
print(list(set([1,2,2,3])))
106. Second largest

python
Copy code
lst=[10,20,30]
print(sorted(set(lst))[-2])
107. Reverse list

python
Copy code
lst=[1,2,3]
lst.reverse()
108. Sort list

python
Copy code
print(sorted([3,1,2]))
109. Merge lists

python
Copy code
print([1,2]+[3,4])
110. List comprehension

python
Copy code
print([x*x for x in range(5)])
111. Even numbers

python
Copy code
print([x for x in range(10) if x%2==0])

112. Odd numbers

python

```
print([x for x in range(10) if x%2])
```

113. Count occurrences

python
Copy code

```
print([1,2,2,3].count(2))
```

114. Remove element

python
Copy code

```
lst=[1,2,3]
lst.remove(2)
```

115. Pop element

python
Copy code

```
lst.pop()
```

116. Flatten list

python
Copy code

```
lst=[[1,2],[3,4]]
print([x for i in lst for x in i])
```

117. Rotate list

python
Copy code

```
lst=[1,2,3,4]
print(lst[2:]+lst[:2])
```

118. Check list empty

python
Copy code

```
print(len([])==0)
```

119. Clone list

python
Copy code

```
new=lst[:]
```

120. Common elements

python
Copy code

```
print(set([1,2,3]) & set([2,3,4]))
```

121. Difference

python
Copy code

```
print(set([1,2,3]) - set([2]))
```

122. Zip lists

python
Copy code

```
print(list(zip([1,2],[3,4])))
```

123. Enumerate

python
Copy code

```
print(list(enumerate(['a','b'])))
```

124. List to tuple

python
Copy code

```
print(tuple([1,2,3]))
```

125. Tuple to list

python
Copy code

```
print(list((1,2,3)))
```

126. Find duplicates

python
Copy code

```
lst=[1,2,2,3]
print({x for x in lst if lst.count(x)>1})
```

127. Index of element

python
Copy code

```
print([1,2,3].index(2))
```

128. Remove all occurrences

python
Copy code

```
lst=[1,2,2,3]
lst=[x for x in lst if x!=2]
```

129. List slicing

python
Copy code

```
print([1,2,3,4][1:3])
```

130. Random shuffle

python
Copy code
import random

lst=[1,2,3]
random.shuffle(lst)
131. Chunk list

python
Copy code
lst=[1,2,3,4]
print([lst[i:i+2] for i in range(0,len(lst),2)])
132. Pair sum

python
Copy code
lst=[1,2,3]
print([(i,j) for i in lst for j in lst if i+j==4])
133. Max occurring

python
Copy code
from collections import Counter
print(Counter([1,2,2,3]).most_common(1))
134. Min occurring

python
Copy code
print(Counter([1,2,2,3]).most_common()[-1])
135. List equality

python
Copy code
print([1,2]==[1,2])
136. Remove None

python
Copy code
print([x for x in [1,None,2] if x])
137. Count even

python
Copy code
print(sum(x%2==0 for x in [1,2,3,4]))

python
Copy code
print(dict(sorted(d.items(),key=lambda x:x[1])))

138. Multiply all

python
Copy code
import math
print(math.prod([1,2,3]))
139. List intersection

python
Copy code
print(list(set([1,2]) & set([2,3])))
140. List union

python
Copy code
print(list(set([1,2]) | set([2,3])))
🔷 DICTIONARY, SET, OOP, FUNCTION (141–200)
141. Create dictionary

python
Copy code
d={'a':1,'b':2}
142. Access value

python
Copy code
print(d['a'])
143. Dictionary keys

python
Copy code
print(d.keys())
144. Dictionary values

python
Copy code
print(d.values())
145. Dictionary items

python
Copy code
print(d.items())
146. Merge dictionaries

python
Copy code
print({**{'a':1},**{'b':2}})

147. Count frequency

python
Copy code
from collections import Counter
print(Counter("hello"))

148. Sort dict by value
print(dict(sorted(d.items(),key=lambda x:x[1])))

149. Remove key

python
Copy code
d.pop('a')
150. Check key exists

python
Copy code
print('a' in d)
151. Set operations

python
Copy code
print({1,2}&{2,3})
152. Frozenset

python
Copy code
fs=frozenset([1,2,3])
153. Lambda

python
Copy code
square=lambda x:x*x
154. Map

python
Copy code
print(list(map(square,[1,2,3])))
155. Filter

python
Copy code
print(list(filter(lambda x:x%2,[1,2,3,4])))

156. Reduce

python
Copy code
from functools import reduce
print(reduce(lambda a,b:a+b,[1,2,3]))
157. Function

python
Copy code
def add(a,b): return a+b
158. Recursive function

python
Copy code
def fact(n): return 1 if n==0 else n*fact(n-1)
159. Generator

python
Copy code
def gen():
  for i in range(3): yield i
160. Iterator

python
Copy code
it=iter([1,2,3])
161. Decorator

python
Copy code
def deco(f):
  def wrap(): print("Hi"); f()
  return wrap
162. Class

python
Copy code
class A: pass
163. Object

python
Copy code
obj=A()
164. Constructor

python
Copy code
class A:
  def __init__(self): print("init")

165. Inheritance

python
Copy code
class B(A): pass
166. Method overriding

python
Copy code
class B(A):
  def show(self): print("B")
167. Encapsulation

python
Copy code
class A:
  def __init__(self): self.__x=10
168. Polymorphism

python
Copy code
print(len("hi"),len([1,2]))
169. Abstraction

python
Copy code
from abc import ABC,abstractmethod
class A(ABC):
  @abstractmethod
  def show(self): pass
170. Static method

python
Copy code
class A:
  @staticmethod
  def show(): print("Hi")
171. Class method

python
Copy code
class A:
  @classmethod
  def show(cls): print(cls)
172. Exception handling

python
Copy code
try: 1/0
except: print("Error")

173. Custom exception

python
Copy code
class MyError(Exception): pass
174. File read

python
Copy code
open("a.txt").read()
175. File write

python
Copy code
open("a.txt","w").write("hi")
176. JSON read

python
Copy code
import json
json.loads('{"a":1}')
177. JSON write

python
Copy code
json.dumps({"a":1})
178. Pickle

python
Copy code
import pickle
pickle.dumps([1,2])
179. Time complexity check

python
Copy code
import time
start=time.time()
180. Thread

python
Copy code
import threading
181. Process

182. Async function
Python

Copy code
async def hello(): pass
183. Await

python
Copy code
await hello()
184. API call

python
Copy code
import requests
requests.get("https://example.com")
185. Virtual env

python
Copy code
# python -m venv env
186. Unit test

python
Copy code
import unittest
187. LRU cache

python
Copy code
from functools import lru_cache
188. Memoization

python
Copy code
@lru_cache(None)
def fib(n): return n if n<2 else fib(n-1)+fib(n-2)
189. Singleton

python
Copy code
class A:
  _i=None
  def __new__(c):
    if not c._i: c._i=super().__new__(c)
    return c._i
190. Factory

python
Copy code
import multiprocessing
191. Observer

python
Copy code
# notify subscribers
192. MVC

python
Copy code
# Model View Controller concept
193. Logging

python
Copy code
import logging
logging.basicConfig(level=logging.INFO)
194. Command line args

python
Copy code
import sys
print(sys.argv)
195. Environment variables

python
Copy code
import os
print(os.getenv("PATH"))
196. OS operations

python
Copy code
import os
os.getcwd()
197. System exit

python
Copy code
import sys
sys.exit()
198. Garbage collection

python
Copy code
import gc
gc.collect()
199. Performance timing

Copy code
def factory(x): return int(x)

import timeit
timeit.timeit("sum(range(10))",number=1000)
200. Python pitfall (mutable default)

python
Copy code
def f(x=[]): x.append(1); return x

python
Copy code

# 50 SQL Interview Queries

1. Find duplicate records in a table

```
SELECT column1, column2, COUNT(*)
FROM your_table
GROUP BY column1, column2
HAVING COUNT(*) > 1;
```

2. Retrieve the second highest salary from the Employee table

```
SELECT MAX(salary) AS
SecondHighestSalary
FROM Employee
WHERE salary < (SELECT MAX(salary)
FROM Employee);
```

3. Find employees without department (Left Join usage)

```
SELECT e.*
FROM Employee e
LEFT JOIN Department d
ON e.department_id =
d.department_id
WHERE d.department_id IS NULL;
```

4. Calculate the total revenue per product

```
SELECT product_id,
SUM(quantity * price) AS
total_revenue
FROM Sales
GROUP BY product_id;
```

5. Get the top 3 highest-paid employees.

```
SELECT TOP 3 *
FROM Employee
ORDER BY salary DESC;
```

6. Customers who made purchases but never returned products.

```
SELECT DISTINCT c.customer_id
FROM Customers c
JOIN Orders o ON c.customer_id =
o.customer_id
WHERE c.customer_id NOT IN (
  SELECT customer_id FROM Returns
);
```

7. Show the count of orders per customer.

```
SELECT customer_id,
COUNT(*) AS order_count
FROM Orders
GROUP BY customer_id;
```

8. Retrieve all employees who joined in 2023.

```
SELECT *
FROM Employee
WHERE YEAR(hire_date) = 2023;
```

9. Calculate the average order value per customer.

```
SELECT customer_id,
AVG(total_amount) AS
avg_order_value
FROM Orders
GROUP BY customer_id;
```

10. Get the latest order placed by each customer.

```
SELECT customer_id,
MAX(order_date) AS
latest_order_date
FROM Orders
GROUP BY customer_id;
```

11. Find products that were never sold.

```
SELECT p.product_id
FROM Products p
LEFT JOIN Sales s
ON p.product_id = s.product_id
WHERE s.product_id IS NULL;
```

12. Identify the most selling product.

```
SELECT TOP 1 product_id,
SUM(quantity) AS total_qty
FROM Sales
GROUP BY product_id
ORDER BY total_qty DESC;
```

13. Get the total revenue and the number of orders per region.

```
SELECT region,
SUM(total_amount) AS total_revenue,
COUNT(*) AS order_count
FROM Orders
GROUP BY region;
```

14. Count how many customers placed more than 5 orders.

```
SELECT COUNT(*) AS customer_count
FROM (
   SELECT customer_id FROM Orders
   GROUP BY customer_id
   HAVING COUNT(*) > 5
) AS subquery;
```

15. Retrieve customers with orders above the average order value.

```
SELECT *
FROM Orders
WHERE total_amount >
(SELECT AVG(total_amount)
FROM Orders);
```

16. Find all employees hired on weekends.

```
SELECT *
FROM Employee
WHERE DATENAME(WEEKDAY, hire_date)
IN ('Saturday', 'Sunday');
```

17. Find all employees hired on weekends.

```
SELECT *
FROM Employee
WHERE EXTRACT(DOW FROM hire_date)
IN (0, 6);
```

18. Get monthly sales revenue and order count.

```
SELECT
FORMAT(date, 'yyyy-MM') AS month,
SUM(amount) AS total_revenue,
COUNT(order_id) AS order_count
FROM Orders
GROUP BY
FORMAT(date, 'yyyy-MM');
```

19. Rank employees by salary within each department.

```
SELECT employee_id, department_id,
salary, RANK() OVER (PARTITION BY
department_id
ORDER BY salary DESC) AS salary_rk
FROM Employee;
```

20. Find customers who placed orders every month in 2023.

```
SELECT customer_id
FROM Orders
WHERE YEAR(order_date) = 2023
GROUP BY customer_id
HAVING COUNT(DISTINCT
FORMAT(order_date,'yyyy-MM')) = 12
```

21. Find moving average of sales over the last 3 days.

```
SELECT order_date,
AVG(total_amount) OVER (ORDER BY
order_date ROWS BETWEEN 2 PRECEDING
AND CURRENT ROW) AS moving_avg
FROM Orders;
```

22. Identify the first and last order date for each customer.

```
SELECT customer_id,
MIN(order_date) AS first_order,
MAX(order_date) AS last_order
FROM Orders
GROUP BY customer_id;
```

23. Show product sales distribution
(percent of total revenue).

```
WITH TotalRevenue AS (
SELECT
SUM(quantity * price) AS total FROM Sales)
SELECT s.product_id,
SUM(s.quantity * s.price) AS revenue,
SUM(s.quantity * s.price) * 100/ t.total
AS revenue_pct
FROM Sales s
CROSS JOIN TotalRevenue t
GROUP BY s.product_id, t.total;
```

24. Retrieve customers who made
consecutive purchases (2 Days)

```
WITH cte AS (
SELECT id, order_date,
LAG(order_date) OVER (PARTITION BY id
ORDER BY order_date) AS prev_order_date
FROM Orders)
SELECT id, order_date, prev_odate
FROM cte
WHERE
DATEDIFF(DAY, prev_odate, order_date) = 1;
```

25. Find churned customers
(no orders in the last 6 months).

```
SELECT customer_id
FROM Orders
GROUP BY customer_id
HAVING
MAX(order_date) <
DATEADD(MONTH,-6,GETDATE());
```

26. Calculate cumulative revenue by
day.

```
SELECT order_date,
SUM(total_amount) OVER
(ORDER BY order_date) AS
cumulative_revenue
FROM Orders;
```

27. Identify top-performing
departments by average salary.

```
SELECT department_id,
AVG(salary) AS avg_salary
FROM Employee
GROUP BY department_id
ORDER BY avg_salary DESC;
```

28. Find customers who ordered
more than the average number of
orders per customer.

```
WITH customer_orders AS (
SELECT customer_id, COUNT(*) AS order_count
FROM Orders
GROUP BY customer_id)
SELECT * FROM customer_orders
WHERE order_count > (SELECT
AVG(order_count) FROM customer_orders);
```

29. Calculate revenue generated from
new customers (first-time orders).

```
WITH first_orders AS (
SELECT customer_id, MIN(order_date) AS
first_order_date FROM Orders
GROUP BY customer_id)
SELECT SUM(o.total_amount) AS new_revenue
FROM Orders o JOIN first_orders f
ON o.customer_id = f.customer_id
WHERE o.order_date = f.first_order_date;
```

30. Find the percentage of
employees in each department.

```
SELECT
  department_id,
  COUNT(*) AS emp_count,
  COUNT(*) * 100.0 / (SELECT
COUNT(*) FROM Employee)
AS pct FROM Employee
GROUP BY department_id;
```

31. Retrieve the maximum salary difference within each department.

```
SELECT
department_id,
MAX(salary) - MIN(salary) AS
salary_diff
FROM Employee
GROUP BY department_id;
```

32. Find products that contribute to 80% of the revenue (Pareto Principle).

```
WITH sales_cte AS (
SELECT product_id, SUM(qty * price) AS revenue
FROM Sales GROUP BY product_id),
total_revenue AS (
SELECT SUM(revenue) AS total FROM sales_cte)
SELECT s.product_id, s.revenue,
SUM(s.revenue) OVER
(ORDER BY s.revenue DESC ROWS BETWEEN UNBOUNDED
PRECEDING AND CURRENT ROW) AS running_total
FROM sales_cte s, total_revenue t
WHERE SUM(s.revenue) OVER (ORDER BY s.revenue DESC
ROWS BETWEEN UNBOUNDED PRECEDING AND
CURRENT ROW) <= t.total * 0.8;
```

33. Calculate average time between two purchases for each customer.

```
WITH cte AS (
SELECT customer_id, order_date,
LAG(order_date) OVER (PARTITION BY
customer_id
ORDER BY order_date) AS prev_date
FROM Orders)
SELECT customer_id,
AVG(DATEDIFF(DAY, prev_date, order_date))
AS avg_gap_days FROM cte
WHERE prev_date IS NOT NULL
GROUP BY customer_id;
```

34. Show last purchase for each customer along with order amount.

```
WITH ranked_orders AS
(SELECT customer_id, order_id,
total_amount, ROW_NUMBER() OVER
(PARTITION BY customer_id ORDER BY
order_date DESC) AS rn FROM Orders)
SELECT customer_id, order_id,
total_amount
FROM ranked_orders
WHERE rn = 1;
```

35. Calculate year-over-year growth in revenue.

```
SELECT FORMAT(order_date, 'yyyy') AS year,
SUM(total_amount) AS revenue,
SUM(total_amount) - LAG(SUM(total_amount))
OVER (ORDER BY FORMAT(order_date, 'yyyy'))
AS yoy_growth
FROM Orders
GROUP BY FORMAT(order_date, 'yyyy');
```

36. Detect customers whose purchase amount is higher than their historical 90th percentile.

```
WITH ranked_orders AS (
SELECT customer_id, order_id,
total_amount,
NTILE(10) OVER (PARTITION BY customer_id
ORDER BY total_amount) AS decile
FROM Orders)
SELECT customer_id, order_id, total_amount
FROM ranked_orders
WHERE decile = 10;
```

37. Find continuous login streaks (e.g., users who logged in 3 or more consecutive days).

```
WITH cte AS (
SELECT user_id, login_date,
DATEDIFF(DAY, ROW_NUMBER() OVER
(PARTITION BY user_id ORDER BY login_date),
login_date) AS grp FROM Logins)
SELECT user_id, MIN(login_date) AS
streak_start, MAX(login_date) AS streak_end,
```

```
COUNT(*) AS streak_length FROM cte
GROUP BY user_id, grp
HAVING COUNT(*) >= 3;
```

38. Calculate customer retention by month (Cohort analysis).

```
WITH Cohorts AS ( SELECT customer_id,
MIN(DATEFROMPARTS(YEAR(order_date),
MONTH(order_date), 1)) AS cohort_month FROM Orders
GROUP BY customer_id),
OrdersByMonth AS (
SELECT customer_id, DATEFROMPARTS(YEAR(order_date),
MONTH(order_date), 1)
AS order_month FROM Orders)
SELECT c.cohort_month, o.order_month,
COUNT(DISTINCT o.customer_id) AS active_customers
FROM Cohorts c
JOIN OrdersByMonth o ON c.customer_id= o.customer_id
GROUP BY c.cohort_month, o.order_month;
```

39. Find products that are always sold together (Market basket analysis).

```
SELECT A.product_id AS product_A,
B.product_id AS product_B,
COUNT(*) AS count_together
FROM Order_Details A
JOIN Order_Details B
ON A.order_id = B.order_id
AND
A.product_id < B.product_id
GROUP BY A.product_id, B.product_id
HAVING COUNT(*) > 10;
```

40. Calculate income inequality (Gini coefficient).

```
WITH income_cte AS (
SELECT salary,
SUM(salary) OVER (ORDER BY salary) AS
cum_incom,
COUNT(*) OVER() AS n,
ROW_NUMBER() OVER (ORDER BY salary) AS r
FROM Employee)
SELECT 1 - (2 * SUM((cum_income) / (SUM(salary)
OVER ()) * (1.0 / n)) ) AS gini_coefficient
FROM income_cte;
```

41. Compute the day when cumulative revenue first exceeded 50% of total revenue (median sales day).

```
WITH cte AS ( SELECT order_date,
SUM(total_amount) AS daily_rev
FROM Orders GROUP BY order_date),
cum_cte AS (
SELECT order_date, daily_rev, SUM(daily_rev) OVER
(ORDER BY order_date) AS cum_rev, SUM(daily_rev)
OVER() AS total_rev FROM cte)
SELECT TOP 1 order_date FROM cum_cte
WHERE cum_rev >= total_rev / 2
ORDER BY order_date;
```

42. Find percentiles (25th, 50th, 75th) of employee salaries.

```
SELECT
(SELECT PERCENTILE_CONT(0.25) WITHIN GROUP
(ORDER BY salary) OVER () FROM Employee) AS p25,
(SELECT PERCENTILE_CONT(0.50) WITHIN GROUP
(ORDER BY salary) OVER () FROM Employee) AS p50,
(SELECT PERCENTILE_CONT(0.75) WITHIN GROUP
(ORDER BY salary) OVER () FROM Employee) AS p75;
```

43. Retrieve customers with increasing order amounts over their last 3 orders.

```
WITH cte AS (
SELECT customer_id, order_date, total_amount,
LAG(total_amount, 2) OVER (PARTITION BY
customer_id ORDER BY order_date) AS amt_t_minus_2,
LAG(total_amount, 1) OVER (PARTITION BY
customer_id ORDER BY order_date)  AS amt_t_minus_1
FROM Orders)
SELECT customer_id, order_date, total_amount
FROM cte
WHERE amt_t_minus_2 < amt_t_minus_1
AND amt_t_minus_1 < total_amount;
```

44. Calculate conversion funnel
between different stages
(e.g., visits →
signups →
purchases).

```
SELECT
SUM(CASE WHEN stage = 'visit' THEN 1
ELSE 0 END) AS visits,
SUM(CASE WHEN stage = 'sign_up' THEN 1
ELSE 0 END) AS sign_ups,
SUM(CASE WHEN stage = 'purchase' THEN 1
ELSE 0 END) AS purchases
FROM Funnel;
```

45. Find the percentage of total sales
contributed by top 10% of customers.

```
WITH cte AS (SELECT customer_id,
SUM(total_amount) AS revenue
FROM Orders GROUP BY customer_id),
ranked AS (SELECT *, NTILE(10) OVER
(ORDER BY revenue DESC) AS decile FROM cte)
SELECT
SUM(revenue) * 100.0 / (SELECT SUM(revenue)
FROM cte) AS pct_top_10
FROM ranked
WHERE decile = 1;
```

46. Calculate weekly active users

```
SELECT DATEPART(YEAR, login_date) AS year,
DATEPART(WEEK, login_date) AS week,
COUNT(DISTINCT user_id) AS wau
FROM Logins
GROUP BY DATEPART(YEAR, login_date),
DATEPART(WEEK, login_date);
```

47. Find employees with salary higher
than department average.

```
WITH dept_avg AS (
SELECT department_id, AVG(salary) AS
avg_salary
FROM Employee
GROUP BY department_id)
SELECT e.* FROM Employee e JOIN dept_avg d
ON e.department_id = d.department_id
WHERE e.salary > d.avg_salary;
```

Found
Helpful ?
Repost
Follow for more!