

## Python Case Study

Q1.

First, we will read the data from the file and load it as JSON

Then we will use pandas.DataFrame and pass the data and the key “people” to turn the JSON data to a Pandas Data Frame

We can also rename the columns we have from the default text file

```
import pandas as pd
import json
filepath = 'EmoplyeeDetails.txt'
with open(filepath, 'r') as file:
    json_data = file.read()
    data = json.loads(json_data)
df = pd.DataFrame(data['people'])
df.rename(columns={'name': 'Name', 'address': 'Address',
                  "mobile_numbers": "Mobile Numbers",
                  "date_of_birth": "DOB"}, inplace=True)
df
```

	Name	Address	Mobile Numbers	DOB
0	John Doe	123 Main St, Cityville, USA	[123-456-7890, 987-654-3210]	1990-05-15
1	Jane Smith	456 Elm St, Townsville, USA	[555-555-5555]	1985-10-25
2	Alice Johnson	789 Oak St, Villagetown, USA	[111-222-3333, 444-555-6666, 777-888-9999]	1978-03-12
3	Bob Brown	987 Pine St, Hamletville, USA	[]	1995-12-03

Q2.

First Let us create Test Files for the code we are going to write.

```
import pandas as pd
import os
import datetime as dt

filepath = 'data'

date_list = []
current_date = dt.datetime.now()
```

```

for i in range(7):
    date_string = current_date.strftime('%Y-%m-%d_%H-%M-%S')
    date_list.append(date_string)
    current_date += dt.timedelta(days=1)

print(date_list)

```

This will create a list of timestamps with the help of the current date and adding 1 day to it. It will create 7 strings with 7 days timestamp for the dummy data.

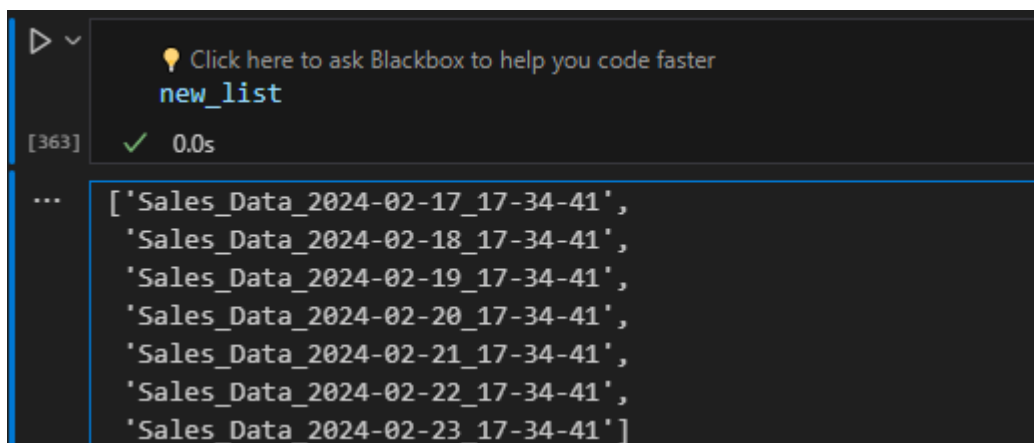
Let's make a new list that will have the full name of the files we want to create.

```

name_str = 'Sales_Data'
new_list = [name_str + '_' + date for date in date_list]

```

This will give us the names that we require for the dummy files.



```

new_list
[363] ✓ 0.0s
... ['Sales_Data_2024-02-17_17-34-41',
     'Sales_Data_2024-02-18_17-34-41',
     'Sales_Data_2024-02-19_17-34-41',
     'Sales_Data_2024-02-20_17-34-41',
     'Sales_Data_2024-02-21_17-34-41',
     'Sales_Data_2024-02-22_17-34-41',
     'Sales_Data_2024-02-23_17-34-41']

```

Once we have the names for the files let us make some dummy data with 7 data points in json format and enter this data into our 7 files

```

json_data = [
    {
        "product": "Widget A",
        "quantity": 100,
        "price_per_unit": 10.50,
        "total_sales": 1050.00
    },
    {
        "product": "Widget B",
        "quantity": 50,
        "price_per_unit": 25.75,
        "total_sales": 1287.50
    }
]

```

```

},
{
    "product": "Widget C",
    "quantity": 75,
    "price_per_unit": 15.00,
    "total_sales": 1125.00
},
{
    "product": "Widget D",
    "quantity": 30,
    "price_per_unit": 50.00,
    "total_sales": 1500.00
},
{
    "product": "Widget E",
    "quantity": 20,
    "price_per_unit": 40.25,
    "total_sales": 805.00
},
{
    "product": "Widget F",
    "quantity": 60,
    "price_per_unit": 18.99,
    "total_sales": 1139.40
},
{
    "product": "Widget G",
    "quantity": 45,
    "price_per_unit": 30.00,
    "total_sales": 1350.00
}
]

import csv
for i in range(len(new_list)):
    filename = os.path.splitext(new_list[i])[0] + ".csv"
    with open(os.path.join(filepath, filename), 'w',
newline='') as f:
        writer = csv.DictWriter(f, fieldnames=["product",
"quantity", "price_per_unit", "total_sales"])
        writer.writeheader()
        writer.writerow(json_data[i])

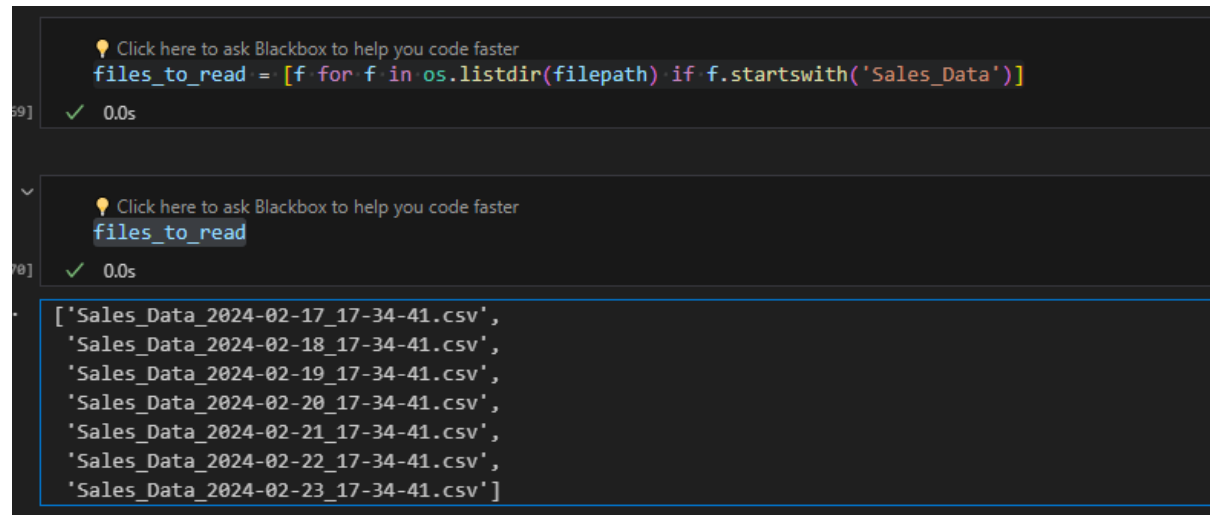
```

Once we get the data loaded and files created, we can now work on our function to load all the data into one table and keep in check if the file data entered is distinct also keeping in mind if data is modified for a previous date to delete and update as required.

First let us try to get all the files that starts with 'Sales\_Data'

```
files_to_read = [f for f in os.listdir(filepath) if f.startswith('Sales_Data')]
```

We can get the names of all the files in the file path using os.listdir .



```
Click here to ask Blackbox to help you code faster
files_to_read = [f for f in os.listdir(filepath) if f.startswith('Sales_Data')]
99] ✓ 0.0s

Click here to ask Blackbox to help you code faster
files_to_read
90] ✓ 0.0s

['Sales_Data_2024-02-17_17-34-41.csv',
'Sales_Data_2024-02-18_17-34-41.csv',
'Sales_Data_2024-02-19_17-34-41.csv',
'Sales_Data_2024-02-20_17-34-41.csv',
'Sales_Data_2024-02-21_17-34-41.csv',
'Sales_Data_2024-02-22_17-34-41.csv',
'Sales_Data_2024-02-23_17-34-41.csv']
```

Once we get the names of the files, we can now get to the next step which is to open all of these and then add the data into a data frame and keep on concatenating the data from different files to the table.

```
def into_table(filepath: str, distinct: set, initial_df:
pd.DataFrame) -> pd.DataFrame:
    try:
        files_to_read = [f for f in os.listdir(filepath) if
f.startswith('Sales_Data')]
        for file in files_to_read:
            date = file.split('_')[2]
            if date not in distinct:
                df = pd.read_csv(os.path.join(filepath, file))
                df['Date'] = date
                initial_df = pd.concat([initial_df, df],
ignore_index=True)
                distinct.add(date)
            else:
                initial_df = initial_df[~initial_df['Date']
.str.startswith(date)]
                df = pd.read_csv(os.path.join(filepath, file))
                df['Date'] = date
                initial_df = pd.concat([initial_df, df],
ignore_index=True)

        return initial_df
    except Exception as e:
        print("An error occurred:", e)
```

```
return initial_df
```

In this function we take an empty data frame as set named distinct and the file path. We then get the names in the list for the files to be read and use a for loop to open and extract the data from the files while checking:

- 1) The date is not in distinct. We open the file using `pd.read_csv` and add the date which we extracted from the title of the file and then we concatenate the df to the initial df and add the date into the set
- 2) If the date is in distinct, we delete the previous data while searching it with the date read the file once again and concatenate the data

This Function However doesn't check if the file is modified and each time just delete's the previous data and concatenate the same data.

To fix this we can use the time we got in our time stamp and check the already added time column with the time we get from the new name of the file.

```
def into_table_modified(filepath: str, distinct: set, initial_df:
pd.DataFrame) -> pd.DataFrame:
    try:
        files_to_read = [f for f in os.listdir(filepath) if
f.startswith('Sales_Data')]
        for file in files_to_read:
            date = file.split('_')[2]
            time = file.split('_')[3].split('.')[0]
            formatted_time = dt.datetime.strptime(time, '%H-%M-
%S').strftime('%H:%M:%S')
            if date not in distinct:
                df = pd.read_csv(os.path.join(filepath, file))
                df['Date'] = date
                df['Time Stamp'] = formatted_time
                initial_df = pd.concat([initial_df, df],
ignore_index=True)
                distinct.add(date)
            else:
                max_time_for_date =
initial_df.loc[initial_df['Date'] == date, 'Time Stamp'].max()
                if formatted_time > max_time_for_date:
                    initial_df =
initial_df[~initial_df['Date'].str.startswith(date)]
                    df = pd.read_csv(os.path.join(filepath, file))
                    df['Date'] = date
                    df['Time Stamp'] = formatted_time
                    initial_df = pd.concat([initial_df, df],
ignore_index=True)

        return initial_df
    except Exception as e:
        print("An error occurred:", e)
```

```
return initial_df
```

Here we have made some changes into the function where we are also checking the previously appended date column with the new date extracted from the file name.

This Fixes all our problems and all the criteria for the problem statement is met.

We can now test the function..

```
distinct = set()
initial_df = pd.DataFrame()
df_new = into_table_modified(filepath, distinct, initial_df)
df_new
```

This is the resultant table we get

	product	quantity	price_per_unit	total_sales	Date	Time Stamp
0	Widget A	100	10.50	1050.0	2024-02-17	17:34:41
1	Widget B	50	25.75	1287.5	2024-02-18	17:34:41
2	Widget C	75	15.00	1125.0	2024-02-19	17:34:41
3	Widget D	30	50.00	1500.0	2024-02-20	17:34:41
4	Widget E	20	40.25	805.0	2024-02-21	17:34:41
5	Widget F	60	18.99	1139.4	2024-02-22	17:34:41
6	Widget G	45	30.00	1350.0	2024-02-23	17:34:41

Now let us add a new file with same date but different time.

```
current_datetime = dt.datetime.now()
date_string = current_datetime.strftime('%Y-%m-%d_%H-%M-%S')
name = 'Sales_Data' + '_' + date_string
json = [ {
    "product": "Widget H",
    "quantity": 42,
    "price_per_unit": 31.00,
    "total_sales": 1250.00
} ]
filename = os.path.splitext(name)[0] + ".csv"
with open(os.path.join(filepath, filename), 'w', newline='')
as f:
```

```

writer = csv.DictWriter(f, fieldnames=["product",
"quantity", "price_per_unit", "total_sales"])
writer.writeheader()
writer.writerow(json[0])

```

Let us run the function again

```
into_table_modified(filepath,distinct,df_new)
```

The resultant data frame is now

	product	quantity	price_per_unit	total_sales	Date	Time Stamp
0	Widget B	50	25.75	1287.5	2024-02-18	17:34:41
1	Widget C	75	15.00	1125.0	2024-02-19	17:34:41
2	Widget D	30	50.00	1500.0	2024-02-20	17:34:41
3	Widget E	20	40.25	805.0	2024-02-21	17:34:41
4	Widget F	60	18.99	1139.4	2024-02-22	17:34:41
5	Widget G	45	30.00	1350.0	2024-02-23	17:34:41
6	Widget H	42	31.00	1250.0	2024-02-17	17:34:42

As you can see the time stamp of the newly and updated data for 17-02-2024 is changed and the function works fine

Q3.

```

def evenPos(string:str) -> str:
    result = ""
    for i in range(0, len(string), 2):
        result += string[i]
    return result

if __name__ == "__main__":
    str = input("Enter a string: ")
    string = evenPos(str)
    print("The string without even posistions are : ",string)

```

```

PS C:\Users\hs414\OneDrive\Desktop\wns> python -u "c:\Users\hs414\OneDrive\Desktop\wns\readStr.py"
Enter a string: Alphabet
The string without even posistions are : Apae

```

## SQL Case Study

Before answering the questions lets make the tables and insert the data we want to add:

```
CREATE DATABASE IF NOT EXISTS Org;
USE Org;

CREATE TABLE IF NOT EXISTS Department (
    Department_id INT UNIQUE NOT NULL PRIMARY KEY,
    Department_name VARCHAR(255)
);
CREATE TABLE IF NOT EXISTS Employee (
    Emp_id INT UNIQUE NOT NULL PRIMARY KEY,
    Emp_Name VARCHAR(255),
    Department_id INT,
    Manager_id INT,
    Salary INT,
    email_Address VARCHAR(255),
    INDEX (Department_id),
    FOREIGN KEY (Department_id) REFERENCES Department(Department_id)
);

-- Inserting data into Department table
INSERT INTO Department (Department_id, Department_name) VALUES
(1, 'IT'),
(3, 'HR'),
(5, 'Payroll');

INSERT INTO Employee (Emp_Name, Emp_id, Department_id, Manager_id, Salary,
email_Address)
VALUES ('Naved', 1, 1, 6, 1000, 'Naved@gmail.com'),
('Atul', 2, 1, 6, 1290, 'Atul@Hotmail.com'),
('Raja', 3, 3, 6, 1500, 'Raja@hotmail.com'),
('Akash', 4, 3, 5, 2000, 'Akash@Outlook.com'),
('Rajan', 6, 5, 6, 30000, 'Rajan@Outlook.com'),
('Balam', 5, 5, 6, 4000, 'Balam@outlook.com'),
('Ravi', 7, 1, 5, 2000, 'Ravi@gmail.com'),
('Ram', 9, 1, 4, 7000, 'Ram@Hotmail.com'),
('Paul', 8, 3, 5, 6000, 'Paul@gmail.com');
```

Now Beginning with the questions



Q1.

```
-- Count of employees in each department

SELECT e.Department_id,d.Department_name,COUNT(e.Emp_id) AS 'Employee Count'
FROM Employee as e
LEFT JOIN Department as d
ON e.Department_id = d.Department_id
GROUP BY e.Department_id,d.department_name;
```

Output:

Department_id	Department_name	Employee Count
1	IT	4
3	HR	3
5	Payroll	2

Q2.

```
-- Highest Salary by Department with Employee Name
SELECT d.Department_name, e.Emp_Name
FROM Department AS d
INNER JOIN(
    SELECT Department_id, MAX(Salary) AS 'Max_Salary'
    FROM Employee
    GROUP BY Department_id
) AS max_sal
ON d.Department_id = max_sal.Department_id
INNER JOIN Employee AS e ON e.Department_id = d.Department_id AND e.Salary =
max_sal.Max_Salary;
```

Output:

Department_name	Emp_Name
Payroll	Rajan
HR	Paul
IT	Ram

Q3.

```
-- Total Salary by Department
SELECT d.department_name, SUM(e.salary) AS 'Total Salary'
FROM Employee as e
JOIN Department as d on e.department_id=d.department_id
GROUP BY d.department_id;

-- Total Salary Paid
SELECT SUM(salary) AS 'Total Salary Paid' FROM Employee;
```

Output:

department_name	Total Salary
IT	11290
HR	9500
Payroll	34000
Total Salary Paid	
	54790

Q4.

```
-- Distinct email domain and their counts
SELECT SUBSTRING_INDEX(email_Address, '@', -1) AS 'Domain', COUNT(*) AS
'Count'
FROM Employee
GROUP BY Domain;
```

Output:

Domain	Count
gmail.com	3
Hotmail.com	3
Outlook.com	3

Q5.

```
-- Name of Employee and its Manager
SELECT e.Emp_Name,m.Emp_Name AS 'Manager'
FROM Employee AS e
LEFT JOIN Employee AS m
ON e.Manager_id = m.Emp_id;
```

Output:

Emp_Name	Manager
Naved	Rajan
Atul	Rajan
Raja	Rajan
Akash	Balam
Balam	Rajan
Rajan	Rajan
Ravi	Balam
Paul	Balam
Ram	Akash

Q6.

```
-- Replica of table without duplicating
CREATE TABLE Employee_Replica LIKE Employee;
```

**EXCEL CASE STUDY**

Q1.

Count Whitespaces = Len of the Sentence – Len of sentence without whitespaces

The quick brown fox jump over the lazy dog
=LEN(A36)-LEN(SUBSTITUTE(A36," ",""))

**=LEN(A36)-LEN (SUBSTITUTE (A36," ",""))**

The quick brown fox jump over the lazy dog
Result
8

Q2.

VLOOKUP is an Excel Named Function that is used to search for things in a column and return the answer of the row that matches

The Syntax is as follows

**=VLOOKUP (Cell, Array/Named Range, Column Number, Exact Match)**

Cell -> The cell to look for

Array -> The array to look from

Col Number -> The column number to look in

Exact Match -> FALSE for exact match

Example: -

A	B
Apple	Red
Bananana	Yellow
Orange	Orange
Grapes	Purple

Let us suppose a table with Fruit names and their colour.

To find the Colour of the Fruit we can use VLOOKUP

**=VLOOKUP (D5, Table3, 2, FALSE)**

Will give:

	Fruit	Result	
	Banana	Yellow	

Q3.

We can check all the named ranges by

1. **Formula Tab → Defined Names → Name Manager**
2. **Pressing CTRL+F3 on keyboard**

Q4.

A)

Surname	Name	Place of Birth	Date of Birth	Gender
Adriana	Deloris	Texas	21-Nov-97	F
Sulivan	Natalie	Oregon	15-Dec-92	F
Sinadra	Vincent	Denver	10-Jun-55	M
Marilyn	Deloris	Denver	24-Dec-76	F
Alberta	Von	Colorado	12-Apr-06	M
Roger	Peter	Miami	10-Jun-65	M
Sedrics	Andrews	Las Vegas	31-Oct-45	M
Keith	Guliver	New Orleans	21-Nov-97	M
Salisbury	Jerry	Chicago	24-Dec-76	M
Aaron	Boris	Oregon	20-Jul-67	M
Yevgnie	Frank	Utah	09-Feb-79	M
Leila	Hanslow	Boston	22-Oct-51	M

First Select any nearby cell and press CTRL+T to create it into a table

Create a new column named Full Name

Use the formula

**=CONCAT (A2, " ", B2)**

Excel will automatically apply it to all the other rows.

After this Select the column and copy it go to:

**Home → Paste → Paste Values**

Now we can remove the first name and last name column from the table.

Full Name	Place of Birth	Date of Birth	Gender
Von Alberta	Colorado	12-04-2006	M
Deloris Adriana	Texas	21-11-1997	F
Guliver Keith	New Orleans	21-11-1997	M
Natalie Sullivan	Oregon	15-12-1992	F
Frank Yevgnie	Utah	09-02-1979	M
Deloris Marilyn	Denver	24-12-1976	F
Jerry Salisbury	Chicago	24-12-1976	M
Boris Aaron	Oregon	20-07-1967	M
Peter Roger	Miami	10-06-1965	M
Vincent Sinadra	Denver	10-06-1955	M
Hanslow Leila	Boston	22-10-1951	M
Andrews Sedrics	Las Vegas	31-10-1945	M

Now for reference lets just enter the filtering options in two separate cells.

Gender	Place of Birth
F	Denver

We can then use the Formula.

**=FILTER (Table2[Full Name], (Table2[Gender]=A17) \* (Table2[Place of Birth] = B17))**

Here the syntax is: -

**=FILTER (Array, Conditions)**

**We use \* To add multiple conditions.**

**Table2[Gender] = A17** means Gender should be equal to value in A17 which is F and **Table2[Place of Birth] = B17** means Place of Birth should be equal to Denver.

This will give us the result as: -

Gender	Place of Birth	Answer
F	Denver	Deloris Marilyn

We can confirm this with our table

Deloris Marilyn	Denver	24-12-1976	F
-----------------	--------	------------	---

B)

Let us approach this step by step

Firstly, let's get the Date of Birth of All the Males Using Filter

**=FILTER (Table2[Date of Birth], Table2[Gender]="M")**

This will give us the Values associated to this filter

Then we can Use LARGE to get the 2 largest years i.e., the second smallest male

**=LARGE (FILTER (Table2[Date of Birth], Table2[Gender]="M"), 2)**

This will give us the date of birth we need to look for

21-11-1997
------------

We can also confirm this by our table by sorting it

12-04-2006	M
21-11-1997	F
21-11-1997	M

We will Then search for the Name using XLOOKUP

**=XLOOKUP (LARGE (FILTER (Table2[Date of Birth], Table2[Gender]="M"),2), Table2[Date of Birth], Table2[Full Name])**

**This will give us our result**

	Result
12-04-2006	Deloris Adriana
21-11-1997	
09-02-1979	
24-12-1976	
20-07-1967	
10-06-1965	
10-06-1955	
22-10-1951	
31-10-1945	

**But since we have 2 people having the same Date of Birth this shows the female candidate**

Now to solve this problem we can use XMATCH to find the row number from down to up in the table it will then give us the correct index to the answer

**=XMATCH (LARGE (FILTER (Table2[Date of Birth], Table2[Gender]="M"), 2), Table2[Date of Birth], 0, -1)**

**This gives us the answer as 3 which is the position of our table in descending order**

Von Alberta	Colorado	12-04-2006	M
Deloris Adriana	Texas	21-11-1997	F
Guliver Keith	New Orleans	21-11-1997	M

We can then Formulate the formula

**=INDEX (XMATCH (LARGE (FILTER (Table2[Date of Birth], Table2 [Gender] = "M"),2), Table2[Date of Birth], 0, -1)**

This gives us our result which is **Guliver Keith**



	Result
12-04-2006	Guliver Keith
21-11-1997	
09-02-1979	
24-12-1976	
20-07-1967	
10-06-1965	
10-06-1955	
22-10-1951	
31-10-1945	

Q5.

First let us create some dummy data to work upon

Name	Salary
Ritik	56000
Hardik	63000
Yash	25000
Shivam	32000
Bhav	46000
Harshit	71000
Jayant	83000

The salary ranges will cover all of our test cases

We will then create references to all the numbers and respected bonuses

Bracket	Bonus
0	0
30000	3000
40000	4000
50000	5000
60000	6000
70000	7000
80000	8000

Once this is done, we can create a new column and write down the formula with nested ifs to find out the bonuses applicable.

**=IFS(AND(B11>\$A\$2,B11<\$A\$3),\$B\$2,AND(B11>\$A\$3,B11<\$A\$4),\$B\$3,AND(B11>\$A\$4,B11<\$A\$5),\$B\$4,AND(B11>\$A\$5,B11<\$A\$6),\$B\$5,AND(B11>\$A\$6,B11<\$A\$7),\$B\$6,AND(B11>\$A\$7,B11<\$A\$8),\$B\$7,B11>\$A\$8,\$B\$8)**

Here we have use absolute referencing so that it works for every column.

Once done we can drag down the formula to the other cells. We will then copy and Paste values so that the results don't change.

We will then get the bonus everyone is going to get.

Name	Salary	Calculated Bonus
Ritik	56000	5000
Hardik	63000	6000
Yash	25000	0
Shivam	32000	3000
Bhav	46000	4000
Harshit	71000	7000
Jayant	83000	8000

Once we get the calculated bonus, we can get the updated salary by:

**=B11+C11**

And drag to rest of the columns.

Name	Salary	Calculated Bonus	New Salary
Ritik	56000	5000	61000
Hardik	63000	6000	69000
Yash	25000	0	25000
Shivam	32000	3000	35000
Bhav	46000	4000	50000
Harshit	71000	7000	78000
Jayant	83000	8000	91000