





import pandas as pd

# Example DataFrame (replace this with your actual data) # Load datasets

employee\_data = pd.read\_csv('employee\_data.csv') compensation\_data = pd.read\_csv('Industry Compensation.csv') ratings\_data = pd.read\_csv('Employee Rating.csv')

# STEP 1: Remove commas from 'Compensation' and convert to numeric employee\_data['Compensation'] = (

employee\_data['Compensation']

.astype(str) # Ensure it's string

.str.replace(',', '', regex=False) # Remove commas

.str.strip() # Remove any leading/trailing spaces

)

# Convert to numeric, setting errors='coerce' to handle any non-numeric strings employee\_data['Compensation'] = pd.to\_numeric(employee\_data['Compensation'], errors= coer

# STEP 2: Fill missing 'Compensation' with median by 'Role' employee\_data['Compensation'] = (

employee\_data.groupby('Role')['Compensation']

.transform(lambda x: x.fillna(x.median()))

)

# STEP 3: Handle 'Years of Experience' (convert and fill missing)

employee\_data['Years of Experience'] = pd.to\_numeric(employee\_data['Years of Experience'] employee\_data['Years of Experience'] = employee\_data['Years of Experience'].fillna(employ

# STEP 4: Check for missing values print("Missing values in cleaned data:") print(employee\_data.isnull().sum())



|  |  |  |  |
| --- | --- | --- | --- |
| Missing values | in | cleaned data: | |
| Employee\_ID |  | 0 | |
| Name |  | 0 | |
| Role |  | 0 | |
| Location |  | 0 | |
| Years of Experience | | | 300 |
| Active? | | | 0 |
| Compensation | | | 0 |
| Last Working Day | | | 263 |

dtype: int64



# Read CSV with proper handling of thousands separator df = pd.read\_csv('employee\_data.csv', thousands=',')

# Clean 'Years of Experience'

df['Years of Experience'] = df['Years of Experience'].replace({ '0-1': 0.5, '01-Feb': 1.1, '02-Mar': 2.2, '03-Apr': 3.3,

'04-May': 4.4, '05-Jun': 5.5, '06-Jul': 6.6, '08-Sep': 8.9

})

df['Years of Experience'] = pd.to\_numeric(df['Years of Experience'], errors='coerce')

# Convert 'Compensation' to integer

df['Compensation'] = pd.to\_numeric(df['Compensation'], errors='coerce')

# Convert 'Active?' to boolean

df['Active?'] = df['Active?'].map({'Y': True, 'N': False})

# Parse 'Last Working Day' as datetime

df['Last Working Day'] = pd.to\_datetime(df['Last Working Day'], errors='coerce', dayfirst

# Optional: Display summary print(df.info())

print(df.head())

<class 'pandas.core.frame.DataFrame'> RangeIndex: 300 entries, 0 to 299 Data columns (total 8 columns):

# Column Non-Null Count Dtype

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 |  | Employee\_ID | 300 | non-null |  | int64 |
| 1 |  | Name | 300 | non-null |  | object |
| 2 |  | Role | 300 | non-null |  | object |
| 3 |  | Location | 300 | non-null |  | object |
| 4 |  | Years of Experience | 289 | non-null |  | float64 |
| 5 |  | Active? | 300 | non-null |  | bool |
| 6 |  | Compensation | 300 | non-null |  | int64 |

7 Last Working Day 33 non-null datetime64[ns]

dtypes: bool(1), datetime64[ns](1), float64(1), int64(2), object(3) memory usage: 16.8+ KB

None

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Employee\_ID | Name | Role | Location | Years | of | Experience \ |
| 0 | 1 | Aditi Khanna | Analyst | Jaipur |  |  | 1.1 |
| 1 | 2 | Pooja Kaur | Senior Analyst | Banglore |  |  | 2.2 |
| 2 | 3 | Kajal Mehta | Analyst | Banglore |  |  | 1.1 |
| 3 | 4 | Amaya Joshi | Analyst | Jaipur |  |  | 0.5 |

4 5 Bilal Kulkarni Associate Jaipur 3.3

Active? Compensation Last Working Day

1. True 653874 NaT
2. True 798703 NaT
3. True 663851 NaT
4. True 570000 NaT
5. True 976125 NaT



import pandas as pd

# Assuming employee\_data is already loaded and processed as you did

employee\_data['Years of Experience'] = pd.to\_numeric(employee\_data['Years of Experience'] employee\_data = employee\_data.dropna(subset=['Years of Experience'])

bins = [0, 1, 2, 5, 10, 20, float('inf')]

labels = ['0-1', '1-2', '2-5', '5-10', '10-20', '20+']

employee\_data['ExperienceRange'] = pd.cut(employee\_data['Years of Experience'], bins=bins

# compensation\_data is loaded (make sure you load it before this) # Rename the compensation column to a consistent name

compensation\_data = compensation\_data.rename(columns={'Average Industry Compensation': 'C

# You need to create 'ExperienceRange' in compensation\_data as well.

# But since compensation\_data likely doesn't have 'Years of Experience', # you should have something to define 'ExperienceRange' there.

# Assuming 'ExperienceRange' already exists or you have a column from which you can creat # If it doesn't exist, you'll need to create it or adjust merge logic.

# If you don't have Years of Experience in compensation\_data but have 'ExperienceRange' a if 'ExperienceRange' not in compensation\_data.columns:

print("Compensation data missing 'ExperienceRange' column, attempting to create it fr if 'Years of Experience' in compensation\_data.columns:

compensation\_data['Years of Experience'] = pd.to\_numeric(compensation\_data['Years compensation\_data = compensation\_data.dropna(subset=['Years of Experience']) compensation\_data['ExperienceRange'] = pd.cut(compensation\_data['Years of Experie

else:

# If no 'Years of Experience' or 'ExperienceRange' in compensation\_data, # you cannot merge on 'ExperienceRange'.

# Consider merging only on 'Role' or adjust accordingly.

print("No 'Years of Experience' or 'ExperienceRange' in compensation\_data; mergin merged = pd.merge(employee\_data, compensation\_data, on='Role', how='left')

else:

# If 'ExperienceRange' exists in compensation\_data, merge on both columns

merged = pd.merge(employee\_data, compensation\_data, on=['Role', 'ExperienceRange'], h

# Rename employee compensation column for clarity if needed

erged = merged.renam olumns Compensation Compensation\_Emp

# Calculate Compensation Gap if both columns exist

if 'Compensation\_Industry' in merged.columns and 'Compensation\_Emp' in merged.columns: merged['CompensationGap'] = merged['Compensation\_Industry'] - merged['Compensation\_Em print(merged.groupby('Role')[['Compensation\_Emp', 'Compensation\_Industry', 'Compensat

else:

print("Cannot calculate CompensationGap because one or both compensation columns are

Compensation data missing 'ExperienceRange' column, attempting to create it from 'Yea No 'Years of Experience' or 'ExperienceRange' in compensation\_data; merging only on ' Empty DataFrame

Columns: [Compensation\_Emp, Compensation\_Industry, CompensationGap] Index: []



# Check actual column names first

print("Columns in employee\_data:", employee\_data.columns)

# Confirm required columns exist

required\_cols = ['Compensation', 'Years of Experience', 'Rating', 'Status'] missing\_cols = [col for col in required\_cols if col not in employee\_data.columns] if missing\_cols:

print(f"Missing columns in employee\_data: {missing\_cols}") else:

# Create 'LeftCompany' column based on 'Status' == 'Inactive' employee\_data['LeftCompany'] = (employee\_data['Status'].str.lower() == 'inactive').as

# Use correct column names, note I replaced 'Experience' with 'Years of Experience' i corr\_cols = ['Compensation', 'Years of Experience', 'Rating', 'LeftCompany']

# Drop rows with NaN in these columns to avoid errors in corr() corr\_data = employee\_data[corr\_cols].dropna()

# Calculate correlation matrix corr = corr\_data.corr()

# Print correlation of all features with 'LeftCompany' sorted descending print(corr['LeftCompany'].sort\_values(ascending=False))

# Group by Status and print average Rating, drop NaN ratings for accuracy print(employee\_data.groupby('Status')['Rating'].mean())

Columns in employee\_data: Index(['Employee\_ID', 'Name', 'Role', 'Location', 'Years of 'Active?', 'Compensation', 'Last Working Day', 'ExperienceRange'], dtype='object')

Missing columns in employee\_data: ['Rating', 'Status']

# 1. Close compensation gaps for roles with high turnover

# 2. Improve retention of employees with low ratings and high experience by upskilling/en # 3. Focus on high turnover locations/roles for management attention

# 4. Consider performance-linked bonuses or stock options for senior staff









# Sample salary structure dataframe: columns = ['Role', 'Compensation', 'Count']

salaries = employee\_data.groupby('Role')['Compensation'].agg(['mean', 'count']).reset\_ind salaries.columns = ['Role', 'AvgCompensation', 'Count']

# Define bonus or stock % for Senior Associate and Manager bonus\_pct = 0.10 # 10% bonus

stock\_pct = 0.15 # 15% stock units

# Filter senior roles

senior\_roles = ['Senior Associate', 'Manager']

senior\_data = salaries[salaries['Role'].isin(senior\_roles)].copy()

# Calculate bonus cost

senior\_data['BonusCost'] = senior\_data['AvgCompensation'] \* bonus\_pct \* senior\_data[ Coun

# Calculate stock cost (assuming stock units valued similarly)

senior\_data['StockCost'] = senior\_data['AvgCompensation'] \* stock\_pct \* senior\_data[ Coun print(senior\_data[['Role', 'BonusCost', 'StockCost']])

# Total impact on PnL

total\_bonus = senior\_data['BonusCost'].sum() total\_stock = senior\_data['StockCost'].sum()

print(f"Total Bonus Cost: INR {total\_bonus:,.0f}") print(f"Total Stock Units Cost: INR {total\_stock:,.0f}")

Empty DataFrame

Columns: [Role, BonusCost, StockCost] Index: []

Total Bonus Cost: INR 0

Total Stock Units Cost: INR 0







import sqlite3

# Connect to SQLite in-memory database (or a file database) conn = sqlite3.connect(':memory:') # or 'your\_database.db' cursor = conn.cursor()

# Create tables with SQL commands cursor.execute('''

CREATE TABLE Employee (

EmployeeID INTEGER PRIMARY KEY,

EmployeeName TEXT, Role TEXT, Location TEXT, Experience REAL, Compensation REAL,

Status TEXT -- 'Active' or 'Inactive'

); ''')

cursor.execute('''

CREATE TABLE IndustryCompensation ( Role TEXT,

ExperienceRange TEXT, AvgCompensation REAL,

PRIMARY KEY (Role, ExperienceRange)

); ''')

cursor.execute('''

CREATE TABLE EmployeeRating ( EmployeeID INTEGER, Rating REAL,

ReviewCycle TEXT,

PRIMARY KEY (EmployeeID, ReviewCycle),

FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

); ''')

conn.commit()

print("Tables created successfully.")

# Don't forget to close connection when done conn.close()

Tables created successfully.





# Add Microsoft repo and key

!curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -

!curl https://packages.microsoft.com/config/ubuntu/20.04/prod.list > /etc/apt/sources.lis

# Update package info

!apt-get update

# Install Microsoft ODBC Driver 17 for SQL Server (accept EULA automatically)

!ACCEPT\_EULA=Y apt-get install -y msodbcsql17

# Install unixODBC development headers (needed by pyodbc)

!apt-get install -y unixodbc-dev

# Install pyodbc Python package

!pip install pyodbc

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| % | Total | % | Received | % | Xferd | Average  Dload | Speed Time  Upload Total | Time  Spent | Time Current  Left Speed |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 --:--:-- | --:--:-- | --:--:-- 0Warning |
| 100 | 983 | 100 | 983 | 0 | 0 | 3104 | 0 --:--:-- | --:--:-- | --: 3110 |
| OK |  |  |  |  |  |  |  |  |  |
| % | Total | % | Received | % | Xferd | Average | Speed Time | Time | Time Current |
|  |  |  |  |  |  | Dload | Upload Total | Spent | Left Speed |
| 100 | 89 | 100 | 89 | 0 | 0 | 416 | 0 --:--:-- | --:--:-- | --: 415 |

Hit:1 [http://security](http://security/) .ubuntu.com/ubuntu jammy-security InRelease Hit:2 <http://archive.ubuntu.com/ubuntu> jammy InRelease

Hit:3 <http://archive.ubuntu.com/ubuntu> jammy-updates InRelease Hit:4 <http://archive.ubuntu.com/ubuntu> jammy-backports InRelease

Hit:5 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86\_64 In Hit:6 https://packages.microsoft.com/ubuntu/20.04/prod focal InRelease

Hit:7 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease Hit:8 https://r2u.stat.illinois.edu/ubuntu jammy InRelease

Hit:9 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease

Hit:10 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease

Hit:11 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease Reading package lists... Done

W: https://packages.microsoft.com/ubuntu/20.04/prod/dists/focal/InRelease: Key is sto W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r Reading package lists... Done

Building dependency tree... Done Reading state information... Done

msodbcsql17 is already the newest version (17.10.6.1-1).

You might want to run 'apt --fix-broken install' to correct these. The following packages have unmet dependencies:

msodbcsql17 : Depends: odbcinst but it is not going to be installed

unixodbc : Depends: odbcinst1debian2 (>= 2.3.11-1) but it is not going to be install Depends: libodbc1 (>= 2.3.11-1) but it is not going to be installed

E: Unmet dependencies. Try 'apt --fix-broken install' with no packages (or specify a Reading package lists... Done

Building dependency tree... Done Reading state information... Done

You might want to run 'apt --fix-broken install' to correct these. The following packages have unmet dependencies:

msodbcsql17 : Depends: odbcinst but it is not going to be installed

unixodbc : Depends: odbcinst1debian2 (>= 2.3.11-1) but it is not going to be install Depends: libodbc1 (>= 2.3.11-1) but it is not going to be installed

unixodbc-dev : Depends: odbcinst1debian2 (= 2.3.11-1) but it is not going to be inst E: Unmet dependencies. Try 'apt --fix-broken install' with no packages (or specify a Requirement already satisfied: pyodbc in /usr/local/lib/python3.11/dist-packages (5.2

import pyodbc

print("Installed ODBC drivers:", pyodbc.drivers())

Installed ODBC drivers: []

driver\_name = 'ODBC Driver 17 for SQL Server'

#Install ODBC driver and pyodbc in Colab (run this cell first)

!curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -

!curl https://packages.microsoft.com/config/ubuntu/20.04/prod.list | sudo tee /etc/apt/so

!sudo apt-get update

!ACCEPT\_EULA=Y sudo apt-get install -y msodbcsql17

!sudo apt-get install -y unixodbc-dev

!pip install pyodbc

# Now restart the runtime (Runtime -> Restart runtime) and then run the following code: import pyodbc

# Check installed drivers after restart print("Available ODBC drivers:", pyodbc.drivers())

driver\_name = ODBC Driver 17 for SQL Server # Confirm this from the output above

server = 'your\_server.database.windows.net' database = 'your\_database'

username = 'your\_username' password = 'your\_password'

conn\_str = (

f'DRIVER={{{driver\_name}}};' f'SERVER={server};' f'DATABASE={database};' f'UID={username};' f'PWD={password}'

)

sql\_create\_proc = """

IF OBJECT\_ID('dbo.FilterEmployees', 'P') IS NOT NULL DROP PROCEDURE dbo.FilterEmployees;

CREATE PROCEDURE dbo.FilterEmployees @Role NVARCHAR(50) = NULL, @Location NVARCHAR(50) = NULL,

@IncludeInactive BIT = 0

AS BEGIN

SET NOCOUNT ON;

END """

try:

SELECT

EmployeeID, EmployeeName, Role, Location, Compensation, Status

FROM Employee

WHERE (@Role IS NULL OR Role = @Role)

AND (@Location IS NULL OR Location = @Location) AND (Status = 'Active' OR @IncludeInactive = 1);

conn = pyodbc.connect(conn\_str) cursor = conn.cursor() cursor.execute(sql\_create\_proc) conn.commit()

print("Stored procedure created successfully.") cursor.close()

conn.close()

except Exception as e: print("Error:", e)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| % | Total | % | Received | % | Xferd | Average Speed Time  Dload Upload Total | Time  Spent | Time Current  Left Speed |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 0 --:--:-- | --:--:-- | --:--:-- 0Warning |
| 100 | 983 | 100 | 983 | 0 | 0 | 4207 0 --:--:-- | --:--:-- | --: 4218 |
| OK |  |  |  |  |  |  |  |  |
| % | Total | % | Received | % | Xferd | Average Speed Time | Time | Time Current |
|  |  |  |  |  |  | Dload Upload Total | Spent | Left Speed |
| 100 | 89 | 100 | 89 | 0 | 0 | 543 0 --:--:-- | --:--:-- | --: 546 |

deb [arch=amd64,armhf,arm64] https://packages.microsoft.com/ubuntu/20.04/prod focal Hit:1 https://packages.microsoft.com/ubuntu/20.04/prod focal InRelease

Get:2 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease [3,632 B] Hit:3 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86\_64 In Hit:4 https://r2u.stat.illinois.edu/ubuntu jammy InRelease

Get:5 [http://security](http://security/) .ubuntu.com/ubuntu jammy-security InRelease [129 kB] Hit:6 <http://archive.ubuntu.com/ubuntu> jammy InRelease

Get:7 <http://archive.ubuntu.com/ubuntu> jammy-updates InRelease [128 kB] Hit:8 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease

Hit:9 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease Hit:10 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease Get:11 <http://archive.ubuntu.com/ubuntu> jammy-backports InRelease [127 kB]

Fetched 388 kB in 2s (201 kB/s) Reading package lists... Done

W: https://packages.microsoft.com/ubuntu/20.04/prod/dists/focal/InRelease: Key is sto W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r Reading package lists... Done

Building dependency tree... Done Reading state information... Done

msodbcsql17 is already the newest version (17.10.6.1-1).

You might want to run 'apt --fix-broken install' to correct these. The following packages have unmet dependencies:

msodbcsql17 : Depends: odbcinst but it is not going to be installed

unixodbc : Depends: odbcinst1debian2 (>= 2.3.11-1) but it is not going to be install Depends: libodbc1 (>= 2.3.11-1) but it is not going to be installed

E: Unmet dependencies. Try 'apt --fix-broken install' with no packages (or specify a Reading package lists... Done

Building dependency tree... Done Reading state information... Done

You might want to run 'apt --fix-broken install' to correct these. The following packages have unmet dependencies:

msodbcsql17 : Depends: odbcinst but it is not going to be installed

unixodbc : Depends: odbcinst1debian2 (>= 2.3.11-1) but it is not going to be install Depends: libodbc1 (>= 2.3.11-1) but it is not going to be installed

unixodbc-dev : Depends: odbcinst1debian2 (= 2.3.11-1) but it is not going to be inst E: Unmet dependencies. Try 'apt --fix-broken install' with no packages (or specify a Requirement already satisfied: pyodbc in /usr/local/lib/python3.11/dist-packages (5.2 Available ODBC drivers: []

Error: ('01000', "[01000] [unixODBC][Driver Manager]Can't open lib 'ODBC Driver 17 fo



!curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -

!curl https://packages.microsoft.com/config/ubuntu/20.04/prod.list > /etc/apt/sources.lis

!apt-get update

!ACCEPT\_EULA=Y apt-get install -y msodbcsql17

!apt-get install -y unixodbc-dev import pyodbc

# Fill these with your actual SQL Server connection info server = 'your\_server.database.windows.net'

database = 'your\_database' username = 'your\_username' password = 'your\_password'

driver = 'ODBC Driver 17 for SQL Server' # Make sure this driver is installed on your ma conn\_str = (

f'DRIVER={{{driver}}};'

f'SERVER={server};' f'DATABASE={database};' f'UID={username};' f'PWD={password}'

)

# SQL to drop the procedure if it exists, then create it sql\_create\_proc = """

IF OBJECT\_ID('dbo.CalculateAverageCompensation', 'P') IS NOT NULL DROP PROCEDURE dbo.CalculateAverageCompensation;

CREATE PROCEDURE dbo.CalculateAverageCompensation @Location NVARCHAR(50) = NULL

AS BEGIN

SET NOCOUNT ON;

END; """

try:

SELECT

Location,

AVG(Compensation) AS AvgCompensation FROM Employee

WHERE (@Location IS NULL OR Location = @Location) AND Status = 'Active'

GROUP BY Location;

# Connect to SQL Server

conn = pyodbc.connect(conn\_str) cursor = conn.cursor()

# Execute the SQL command to create the procedure cursor.execute(sql\_create\_proc)

conn.commi

print("Stored procedure 'CalculateAverageCompensation' created successfully.") except Exception as e:

print("Error:", e)

finally:

# Close connections cleanly try:

cursor.close() except:

pass try:

conn.close() except:

pass

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| % | Total | % | Received | % | Xferd | Average Speed Time  Dload Upload Total | Time  Spent | Time Current  Left Speed |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 0 --:--:-- | --:--:-- | --:--:-- 0Warning |
| 100 | 983 | 100 | 983 | 0 | 0 | 4824 0 --:--:-- | --:--:-- | --: 4842 |
| OK |  |  |  |  |  |  |  |  |
| % | Total | % | Received | % | Xferd | Average Speed Time | Time | Time Current |
|  |  |  |  |  |  | Dload Upload Total | Spent | Left Speed |
| 100 | 89 | 100 | 89 | 0 | 0 | 357 0 --:--:-- | --:--:-- | --: 358 |

Hit:1 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease

Hit:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86\_64 In Hit:3 https://packages.microsoft.com/ubuntu/20.04/prod focal InRelease

Hit:4 https://r2u.stat.illinois.edu/ubuntu jammy InRelease

Hit:5 [http://security](http://security/) .ubuntu.com/ubuntu jammy-security InRelease Hit:6 <http://archive.ubuntu.com/ubuntu> jammy InRelease

Hit:7 <http://archive.ubuntu.com/ubuntu> jammy-updates InRelease Hit:8 <http://archive.ubuntu.com/ubuntu> jammy-backports InRelease

Hit:9 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease

Hit:10 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease Hit:11 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease Reading package lists... Done

W: https://packages.microsoft.com/ubuntu/20.04/prod/dists/focal/InRelease: Key is sto W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r Reading package lists... Done

Building dependency tree... Done Reading state information... Done

msodbcsql17 is already the newest version (17.10.6.1-1).

You might want to run 'apt --fix-broken install' to correct these. The following packages have unmet dependencies:

msodbcsql17 : Depends: odbcinst but it is not going to be installed

unixodbc : Depends: odbcinst1debian2 (>= 2.3.11-1) but it is not going to be install Depends: libodbc1 (>= 2.3.11-1) but it is not going to be installed

E: Unmet dependencies. Try 'apt --fix-broken install' with no packages (or specify a Reading package lists... Done

Building dependency tree... Done Reading state information... Done

You might want to run 'apt --fix-broken install' to correct these. The following packages have unmet dependencies:

msodbcsql17 : Depends: odbcinst but it is not going to be installed

unixodbc : Depends: odbcinst1debian2 (>= 2.3.11-1) but it is not going to be install Depends: libodbc1 (>= 2.3.11-1) but it is not going to be installed

unixodbc-dev : Depends: odbcinst1debian2 (= 2.3.11-1) but it is not going to be inst E: Unmet dependencies. Try 'apt --fix-broken install' with no packages (or specify a Error: ('01000', "[01000] [unixODBC][Driver Manager]Can't open lib 'ODBC Driver 17 fo





!pip install ipython-sql

%load\_ext sql

%sql sqlite://

Requirement already satisfied: ipython-sql in /usr/local/lib/python3.11/dist-packages Requirement already satisfied: prettytable in /usr/local/lib/python3.11/dist-packages Requirement already satisfied: ipython in /usr/local/lib/python3.11/dist-packages (fr Requirement already satisfied: sqlalchemy>=2.0 in /usr/local/lib/python3.11/dist-pack Requirement already satisfied: sqlparse in /usr/local/lib/python3.11/dist-packages (f Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (from i Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.11/dist-pac Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.11/dist-packages Requirement already satisfied: typing-extensions>=4.6.0 in /usr/local/lib/python3.11/ Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.11/dist-pac Collecting jedi>=0.16 (from ipython->ipython-sql)

Downloading jedi-0.19.2-py2.py3-none-any.whl.metadata (22 kB)

Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages ( Requirement already satisfied: pickleshare in /usr/local/lib/python3.11/dist-packages Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.11/dist-packa Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/l Requirement already satisfied: pygments in /usr/local/lib/python3.11/dist-packages (f Requirement already satisfied: backcall in /usr/local/lib/python3.11/dist-packages (f Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.11/dist-pa Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dist-packages Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (fr Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.11/dist- Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.11/dist-pack Downloading jedi-0.19.2-py2.py3-none-any.whl (1.6 MB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 16.8 MB/s eta 0:00:00 Installing collected packages: jedi

Successfully installed jedi-0.19.2

%%sql

DROP TABLE IF EXISTS Employees;

CREATE TABLE Employees (

EmployeeID INTEGER PRIMARY KEY AUTOINCREMENT,

EmployeeName TEXT, Role TEXT, Location TEXT, Compensation REAL, Experience REAL, Status TEXT

);

INSERT INTO Employees (EmployeeName, Role, Location, Compensation, Experience, Status) VA ('Alice Smith', 'Consultant', 'Bangalore', 90000, 3.5, 'Active'),

('Bob Jones', 'Analyst', 'Mumbai', 75000, 2, 'Active'),

('Carol Lee', 'Manager', 'Delhi', 120000, 7, 'Inactive'), ('David Kim', 'Consultant', 'Bangalore', 95000, 4, 'Active'), ('Eva Zhang', 'Analyst', 'Mumbai', 72000, 1, 'Active');

\* sqlite:// Done.

Done.

1. rows affected. []

import sqlite3

conn = sqlite3.connect(':memory:') # In-memory DB cur = conn.cursor()

cur.execute('DROP TABLE IF EXISTS Employees;') cur.execute('''

CREATE TABLE Employees (

EmployeeID INTEGER PRIMARY KEY AUTOINCREMENT,

EmployeeName TEXT, Role TEXT, Location TEXT, Compensation REAL, Experience REAL, Status TEXT

) ''')

employees = [

('Alice Smith', 'Consultant', 'Bangalore', 90000, 3.5, 'Active'), ('Bob Jones', 'Analyst', 'Mumbai', 75000, 2, 'Active'),

('Carol Lee', 'Manager', 'Delhi', 120000, 7, 'Inactive'), ('David Kim', 'Consultant', 'Bangalore', 95000, 4, 'Active'), ('Eva Zhang', 'Analyst', 'Mumbai', 72000, 1, 'Active')

]

cur.executemany('INSERT INTO Employees (EmployeeName, Role, Location, Compensation, Exper conn.commit()

# Query example:

cur.execute("SELECT \* FROM Employees") rows = cur.fetchall()

for row in rows: print(row)

conn.close()

(1, 'Alice Smith', 'Consultant', 'Bangalore', 90000.0, 3.5, 'Active')

(2, 'Bob Jones', 'Analyst', 'Mumbai', 75000.0, 2.0, 'Active')

(3, 'Carol Lee', 'Manager', 'Delhi', 120000.0, 7.0, 'Inactive')

(4, 'David Kim', 'Consultant', 'Bangalore', 95000.0, 4.0, 'Active')

(5, 'Eva Zhang', 'Analyst', 'Mumbai', 72000.0, 1.0, 'Active')

import sqlite3 import pandas as pd

def create\_database():

conn = sqlite3.connect(':memory:') cur = conn.cursor()

# Drop table if exists

cur.execute('DROP TABLE IF EXISTS Employees;')

# Create table cur.execute('''

CREATE TABLE Employees (

EmployeeID INTEGER PRIMARY KEY AUTOINCREMENT,

EmployeeName TEXT, Role TEXT, Location TEXT, Compensation REAL, Experience REAL, Status TEXT

) ''')

# Insert sample data - extend as needed employees = [

('Alice Smith', 'Consultant', 'Bangalore', 90000, 3.5, 'Active'), ('Bob Jones', 'Analyst', 'Mumbai', 75000, 2, 'Active'),

('Carol Lee', 'Manager', 'Delhi', 120000, 7, 'Inactive'), ('David Kim', 'Consultant', 'Bangalore', 95000, 4, 'Active'), ('Eva Zhang', 'Analyst', 'Mumbai', 72000, 1, 'Active'),

Frank Liu Senior Associate Delhi 110000 6 Inactive )

('Grace Park', 'Manager', 'Mumbai', 125000, 8, 'Active'),

('Hannah Scott', 'Consultant', 'Bangalore', 91000, 3, 'Inactive'), ('Ian Wright', 'Senior Associate', 'Mumbai', 108000, 5, 'Active'), ('Judy Chan', 'Analyst', 'Delhi', 68000, 2, 'Inactive'),

]

cur.executemany('''

INSERT INTO Employees (EmployeeName, Role, Location, Compensation, Experience, St VALUES (?, ?, ?, ?, ?, ?)

''', employees)

conn.commit() return conn, cur

def data\_quality\_check(cur):

print("=== Data Quality Checks ===")

# Check for null or missing values in any column cur.execute("""

SELECT COUNT(\*) FROM Employees

WHERE EmployeeName IS NULL OR Role IS NULL OR Location IS NULL OR Compensation IS NULL OR Experience IS NULL OR Status IS NULL

""")

missing\_count = cur.fetchone()[0] if missing\_count > 0:

print(f"Data Quality Issue: {missing\_count} records have missing values.") else:

print("No missing values detected.")

def turnover\_by\_location\_and\_role(cur):

print("\n=== Turnover Analysis by Location and Role ===")

# Turnover = count of inactive employees / total employees in that group query = '''

SELECT Location, Role,

COUNT(\*) AS TotalEmployees,

SUM(CASE WHEN Status = 'Inactive' THEN 1 ELSE 0 END) AS TurnoverCount, ROUND(100.0 \* SUM(CASE WHEN Status = 'Inactive' THEN 1 ELSE 0 END) / COUNT(\*),

FROM Employees

GROUP BY Location, Role

ORDER BY TurnoverPercent DESC '''

cur.execute(query) rows = cur.fetchall()

df = pd.DataFrame(rows, columns=['Location', 'Role', 'TotalEmployees', TurnoverCount print(df)

return df

def compensation\_vs\_industry(cur, industry\_comp): print("\n=== Compensation vs Industry Standards ===") query = '''

SELECT Role,

ROUND(AVG(Compensation), 2) AS AvgCompensation, ROUND(AVG(Experience), 2) AS AvgExperience

FROM Employees GROUP BY Role '''

cur.execute(query) rows = cur.fetchall()

df\_emp = pd.DataFrame(rows, columns=['Role', 'AvgCompensation', 'AvgExperience']) df = pd.merge(df\_emp, industry\_comp, on='Role', how='left')

df['Compensation vs Industry (%)'] = round((df['AvgCompensation'] / df['IndustryCompe print(df)

return df

def analyze\_headcount\_decline():

print("\n=== Main Reasons for Headcount Decline ===") print("""

* Higher turnover rates in certain locations and roles.
* Compensation for some roles might be below industry standards.
* Senior roles lack additional incentives such as bonuses or stock units.
* Possible lack of growth opportunities and employee engagement.
* Presence of inactive employees reducing active headcount. """)

def recommend\_retention\_strategies():

print("\n=== Recommendations to Retain Employees ===") print("""

1. Align compensation with or exceed industry benchmarks.
2. Introduce bonuses or stock units for senior roles (Senior Associate, Manager).
3. Enhance career development and training programs.
4. Improve employee engagement, work-life balance, and recognition.
5. Regular feedback and performance review processes. """)

def financial\_impact\_analysis(cur):

print("\n=== Financial Impact Analysis: Bonuses vs Stock Units ===") query = '''

SELECT Role, Compensation FROM Employees

WHERE Role IN ('Senior Associate', 'Manager') '''

cur.execute(query) rows = cur.fetchall()

df = pd.DataFrame(rows, columns=['Role', 'Compensation'])

if df.empty:

print("No Senior Associate or Manager data available for financial impact analysi return

total\_comp = df['Compensation'].sum() bonus\_rate = 0.10 # 10% bonus assumed

stock\_units\_rate = 0.15 # 15% stock units assumed value

bonus\_cost = total\_comp \* bonus\_rate stock\_units\_cost = total\_comp \* stock\_units\_rate

print(f"Total current compensation of senior roles: INR {total\_comp:,.2f}") print(f"Estimated additional cost with 10% Bonuses: INR {bonus\_cost:,.2f}") print(f"Estimated additional cost with 15% Stock Units: INR {stock\_units\_cost:,.2f}")

recommendation = "Bonus" if bonus\_cost < stock\_units\_cost else "Stock Units" print(f"Recommendation: Implement {recommendation} based on financial impact.")

def stakeholder\_impact\_and\_implementation():

print("\n=== Stakeholders and Implementation Recommendations ===") print("""

Stakeholders:

* Senior Associates and Managers (direct impact)
* HR and Compensation teams
* Finance department
* Executive leadership
* All employees (indirect impact)

Implementation Steps:

1. Communicate changes transparently to employees.
2. Obtain budget approval from finance.
3. Phase-wise rollout and monitor impact.
4. Train HR and managers on new compensation structures.
5. Collect feedback and adjust accordingly. """)

def export\_to\_excel(conn):

print("\nExporting employee data to 'techsolve\_employee\_data.xlsx' ...") df = pd.read\_sql\_query("SELECT \* FROM Employees", conn) df.to\_excel("techsolve\_employee\_data.xlsx", index=False)

print("Export completed.")

def main():

industry\_comp = pd.DataFrame({

'Role': ['Analyst', 'Consultant', 'Manager', 'Senior Associate'], 'IndustryCompensation': [70000, 92000, 125000, 110000]

})

conn, cur = create\_database() data\_quality\_check(cur) turnover\_by\_location\_and\_role(cur) compensation\_vs\_industry(cur, industry\_comp) analyze\_headcount\_decline() recommend\_retention\_strategies() financial\_impact\_analysis(cur) stakeholder\_impact\_and\_implementation() export\_to\_excel(conn)

conn.close()

if name == " main ": main()

=== Data Quality Checks === No missing values detected.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| === | Turnover  Location | Analysis by Location and Role ===  Role TotalEmployees | TurnoverCount | TurnoverPercent |
| 0 | Delhi | Analyst 1 | 1 | 100.00 |
| 1 | Delhi | Manager 1 | 1 | 100.00 |
| 2 | Delhi | Senior Associate 1 | 1 | 100.00 |
| 3 | Bangalore | Consultant 3 | 1 | 33.33 |
| 4 | Mumbai | Analyst 2 | 0 | 0.00 |
| 5 | Mumbai | Manager 1 | 0 | 0.00 |
| 6 | Mumbai | Senior Associate 1 | 0 | 0.00 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| === | Compensation vs  Role | Industry Standards  AvgCompensation | ===  AvgExperience | IndustryCompensation | \ |
| 0 | Analyst | 71666.67 | 1.67 | 70000 |  |
| 1 | Consultant | 92000.00 | 3.50 | 92000 |  |
| 2 | Manager | 122500.00 | 7.50 | 125000 |  |
| 3 | Senior Associate | 109000.00 | 5.50 | 110000 |  |

Compensation vs Industry (%)

0 102.38

1 100.00

2 98.00

3 99.09

=== Main Reasons for Headcount Decline ===

* Higher turnover rates in certain locations and roles.
* Compensation for some roles might be below industry standards.
* Senior roles lack additional incentives such as bonuses or stock units.
* Possible lack of growth opportunities and employee engagement.
* Presence of inactive employees reducing active headcount.

=== Recommendations to Retain Employees ===

1. Align compensation with or exceed industry benchmarks.
2. Introduce bonuses or stock units for senior roles (Senior Associate, Manager).
3. Enhance career development and training programs.
4. Improve employee engagement, work-life balance, and recognition.
5. Regular feedback and performance review processes.

=== Financial Impact Analysis: Bonuses vs Stock Units === Total current compensation of senior roles: INR 463,000.00 Estimated additional cost with 10% Bonuses: INR 46,300.00 Estimated additional cost with 15% Stock Units: INR 69,450.00 Recommendation: Implement Bonus based on financial impact.

=== Stakeholders and Implementation Recommendations === Stakeholders:

* + Senior Associates and Managers (direct impact)
  + HR and Compensation teams
  + Finance department
  + Executive leadership
  + All employees (indirect impact)



import sqlite3

# Create in-memory SQLite DB

conn = sqlite3.connect(':memory:') cur = conn.cursor()

# Create table cur.execute('''

CREATE TABLE Employees (

EmployeeID INTEGER PRIMARY KEY AUTOINCREMENT,

EmployeeName TEXT, Role TEXT, Location TEXT, Compensation REAL, Experience REAL, Status TEXT

) ''')

# Insert sample data

cur.execute("INSERT INTO Employees (EmployeeName, Role, Location, Compensation, Experienc conn.commit()

# Query data

cur.execute("SELECT \* FROM Employees") rows = cur.fetchall()

for row in rows: print(row)

conn.close()

(1, 'John Doe', 'Manager', 'NY', 70000.0, 5.0, 'Active')

import sqlite3

# Create connection to in-memory DB conn = sqlite3.connect(':memory:') cur = conn.cursor()

# Create Employees table cur.execute('''

CREATE TABLE Employees (

EmployeeID INTEGER PRIMARY KEY AUTOINCREMENT,

EmployeeName TEXT, Role TEXT, Location TEXT, Compensation REAL, Experience REAL, Status TEXT

) ''')

# Insert sample employees employees = [

('John Doe', 'Manager', 'New York', 70000, 5, 'Active'),

('Jane Smith', 'Developer', 'Chicago', 65000, 3, 'Active'), ('Mike Johnson', 'Manager', 'New York', 72000, 6, 'Inactive'), ('Sara Lee', 'Developer', 'Seattle', 68000, 4, 'Active')

]

cur.executemany('INSERT INTO Employees (EmployeeName, Role, Location, Compensation, Exper conn.commit()

# Example: Filter active managers in New York role = 'Manager'

location = 'New York' include\_inactive = False

query = '''

SELECT EmployeeID, EmployeeName, Role, Location, Compensation, Experience, Status FROM Employees

WHERE (Role = ? OR ? IS NULL) AND (Location = ? OR ? IS NULL)

AND (Status = 'Active' OR (? = 1 AND Status = 'Inactive')) '''

cur.execute(query, (role, role, location, location, 1 if include\_inactive else 0)) rows = cur.fetchall()

print("Filtered Employees:") for row in rows:

print(row)

# Example: Simulate increment by 10% for all employees increment\_percent = 10

cur.execute('''

SELECT EmployeeID, EmployeeName, Role, Location, Compensation,

Compensation \* (1 + ? / 100.0) AS NewCompensation, Experience, Status FROM Employees

''', (increment\_percent,))

print("\nCompensation Increment Simulation:") for row in cur.fetchall():

print(row) conn.close()

Filtered Employees:

(1, 'John Doe', 'Manager', 'New York', 70000.0, 5.0, 'Active')

Compensation Increment Simulation:

|  |  |  |
| --- | --- | --- |
| (1, | 'John | Doe', 'Manager', 'New York', 70000.0, 77000.0, 5.0, 'Active') |
| (2, | 'Jane | Smith', 'Developer', 'Chicago', 65000.0, 71500.0, 3.0, 'Active') |
| (3, | 'Mike | Johnson', 'Manager', 'New York', 72000.0, 79200.0, 6.0, 'Inactive') |
| (4, | 'Sara | Lee', 'Developer', 'Seattle', 68000.0, 74800.0, 4.0, 'Active') |

!pip install pyodbc pandas

Requirement already satisfied: pyodbc in /usr/local/lib/python3.11/dist-packages (5.2 Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2 Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packag Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/di Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-package Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packa Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (f

class Employee:

def init (self, EmployeeID, EmployeeName, Role, Location, Compensation, Experience self.EmployeeID = EmployeeID

self.EmployeeName = EmployeeName self.Role = Role

self.Location = Location self.Compensation = Compensation self.Experience = Experience self.Status = Status

class LocationCompensation:

def init (self, Location, AvgCompensation): self.Location = Location self.AvgCompensation = AvgCompensation

EmployeeCompensationSimulation Employee

def init (self, EmployeeID, EmployeeName, Role, Location, Compensation, Experience super(). init (EmployeeID, EmployeeName, Role, Location, Compensation, Experien self.NewCompensation = NewCompensation

import pandas as pd

def filter\_employees(role=None, location=None, include\_inactive=False): query = "EXEC FilterEmployees @Role=?, @Location=?, @IncludeInactive=?" params = (role, location, 1 if include\_inactive else 0)

return pd.read\_sql(query, conn, params=params)

def calculate\_average\_compensation(location=None):

query = "EXEC CalculateAverageCompensation @Location=?" return pd.read\_sql(query, conn, params=(location,))

def simulate\_increment(increment\_percent, employee\_id=None, location=None):

query = "EXEC SimulateIncrement @IncrementPercent=?, @EmployeeID=?, @Location=?" return pd.read\_sql(query, conn, params=(increment\_percent, employee\_id, location))

def export\_to\_csv(df, filename='simulated\_compensation.csv'): df.to\_csv(filename, index=False)

print(f"Saved to {filename}")

import pandas as pd

# Sample employee data employees = pd.DataFrame([

{'EmployeeID': 1, 'EmployeeName': 'Alice', 'Role': 'Analyst', 'Location': 'NY', 'Comp

{'EmployeeID': 2, 'EmployeeName': 'Bob', 'Role': 'Manager', 'Location': 'NY', 'Compen

{'EmployeeID': 3, 'EmployeeName': 'Charlie', 'Role': 'Analyst', 'Location': 'SF', 'Co

])

def export\_to\_csv(df, filename='simulated\_compensation.csv'): df.to\_csv(filename, index=False)

print(f"Saved to: {filename}")

def simulate\_increment(increment\_percent, df=employees): df = df.copy()

df['NewCompensation'] = df['Compensation'] \* (1 + increment\_percent / 100)

df\_simulation = simulate\_increment(10) # 10% increment export\_to\_csv(df\_simulation)

df\_simulation # Show output in Colab

Saved to: simulated\_compensation.csv

**EmployeeID EmployeeName Role Location Compensation Experience Status New**

1. 1 Alice Analyst NY 60000 3 Active
2. 2 Bob Manager NY 80000 5 Active
3. 3 Charlie Analyst SF 65000 4 Inactive

# For Colab only: install ODBC driver

!sudo apt-get install unixodbc-dev

!pip install pyodbc

Reading package lists... Done Building dependency tree... Done Reading state information... Done

You might want to run 'apt --fix-broken install' to correct these. The following packages have unmet dependencies:

msodbcsql17 : Depends: odbcinst but it is not going to be installed

unixodbc : Depends: odbcinst1debian2 (>= 2.3.11-1) but it is not going to be install Depends: libodbc1 (>= 2.3.11-1) but it is not going to be installed

unixodbc-dev : Depends: odbcinst1debian2 (= 2.3.11-1) but it is not going to be inst E: Unmet dependencies. Try 'apt --fix-broken install' with no packages (or specify a Requirement already satisfied: pyodbc in /usr/local/lib/python3.11/dist-packages (5.2

# Step 1: Install dependencies (Run once)

!sudo apt-get install -y unixodbc-dev

!pip install pyodbc

Reading package lists... Done Building dependency tree... Done Reading state information... Done

You might want to run 'apt --fix-broken install' to correct these. The following packages have unmet dependencies:

msodbcsql17 : Depends: odbcinst but it is not going to be installed

unixodbc : Depends: odbcinst1debian2 (>= 2.3.11-1) but it is not going to be install Depends: libodbc1 (>= 2.3.11-1) but it is not going to be installed

unixodbc-dev : Depends: odbcinst1debian2 (= 2.3.11-1) but it is not going to be inst E: Unmet dependencies. Try 'apt --fix-broken install' with no packages (or specify a Requirement already satisfied: pyodbc in /usr/local/lib/python3.11/dist-packages (5.2

# Step 1: Install Microsoft ODBC Driver 17 for SQL Server & dependencies (Colab-compatibl

!curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -

!sudo curl https://packages.microsoft.com/config/ubuntu/20.04/prod.list | sudo tee /etc/a

!sudo apt-get update

!sudo ACCEPT\_EULA=Y apt-get install -y msodbcsql17

!sudo apt-get install -y unixodbc-dev

!pip install pyodbc

# Step 2: Connect to SQL Server using pyodbc (replace placeholders with your actual conn import pyodbc

server = 'your\_server.database.windows.net' # e.g., 'sqlserver123.database.windows.net' database = 'your\_database' # e.g., 'EmployeeDB'

username = 'your\_username' # e.g., 'adminuser'

password = 'your\_password' # e.g., 'StrongPassword123'

try:

conn = pyodbc.connect(

f"DRIVER={{ODBC Driver 17 for SQL Server}};" f"SERVER={server};"

f"DATABASE={database};" f"UID={username};" f"PWD={password};" "TrustServerCertificate=Yes;" "Encrypt=yes;"

)

cursor = conn.cursor()

print(" Connected successfully to SQL Server!") except Exception as e:

print(" Connection failed:", e)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| % Total % Received % Xferd | Average Speed | Time | Time | Time | Current |
|  | Dload Upload | Total | Spent | Left | Speed |

100 983 100 983 0 0 3906 0 --:--:-- --:--:-- --: 3916

Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see ap OK

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| % | Total % Received % Xferd | Average Speed | Time | Time | Time Current |
|  |  | Dload Upload | Total | Spent | Left Speed |
| 100 | 89 100 89 0 0 692 0 --:--:-- --:--:-- --: 689 | | | | |
| deb | [arch=amd64,armhf,arm64] https://packages.microsoft.com/ubuntu/20.04/prod focal | | | | |

Get:1 [http://security](http://security/) .ubuntu.com/ubuntu jammy-security InRelease [129 kB]

Get:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86\_64 InR Get:3 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease [3,632 B] Get:4 https://packages.microsoft.com/ubuntu/20.04/prod focal InRelease [3,632 B] Hit:5 <http://archive.ubuntu.com/ubuntu> jammy InRelease

Get:6 https://r2u.stat.illinois.edu/ubuntu jammy InRelease [6,555 B]

Get:7 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86\_64 Pac Get:8 <http://archive.ubuntu.com/ubuntu> jammy-updates InRelease [128 kB]

Hit:9 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease

Hit:10 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease Get:11 <http://archive.ubuntu.com/ubuntu> jammy-backports InRelease [127 kB]

Get:12 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 Packages [35

Get:13 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease [24.6 kB Get:14 https://packages.microsoft.com/ubuntu/20.04/prod focal/main all Packages [2,93 Get:15 https://packages.microsoft.com/ubuntu/20.04/prod focal/main arm64 Packages [79 Get:16 https://packages.microsoft.com/ubuntu/20.04/prod focal/main armhf Packages [26 Get:17 [http://security](http://security/) .ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [4, Get:18 [http://security](http://security/) .ubuntu.com/ubuntu jammy-security/main amd64 Packages [2,934 kB Get:19 https://r2u.stat.illinois.edu/ubuntu jammy/main amd64 Packages [2,725 kB] Get:20 https://r2u.stat.illinois.edu/ubuntu jammy/main all Packages [8,944 kB]

Get:21 <http://archive.ubuntu.com/ubuntu> jammy-updates/universe amd64 Packages [1,546 Get:22 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 Packages [3,245 kB] Get:23 <http://archive.ubuntu.com/ubuntu> jammy-updates/restricted amd64 Packages [4,51 Get:24 <http://archive.ubuntu.com/ubuntu> jammy-backports/universe amd64 Packages [35.2 Get:25 <http://archive.ubuntu.com/ubuntu> jammy-backports/main amd64 Packages [83.2 kB] Get:26 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy/main amd64 Package Fetched 31.0 MB in 4s (7,103 kB/s)

Reading package lists... Done

W: https://packages.microsoft.com/ubuntu/20.04/prod/dists/focal/InRelease: Key is sto W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r Reading package lists... Done

Building dependency tree... Done Reading state information... Done

The following additional packages will be installed: libodbc1 odbcinst odbcinst1debian2 unixodbc

Suggested packages: unixodbc-bin

The following NEW packages will be installed:

libodbc1 msodbcsql17 odbcinst odbcinst1debian2 unixodbc

0 upgraded, 5 newly installed, 0 to remove and 90 not upgraded. Need to get 1,404 kB of archives.

After this operation, 1,059 kB of additional disk space will be used.

Get:1 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 libodbc1 amd6 Get:2 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 odbcinst1debi Get:3 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 odbcinst amd6 Get:4 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 unixodbc amd6 Get:5 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 msodbcsql17 a Fetched 1,404 kB in 0s (4,260 kB/s)

debconf: unable to initialize frontend: Dialog

# Run this first to set up the DataFrame import pandas as pd

# Sample Employee Dataset data = {

'EmployeeID': range(1, 11),

'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace', 'Heidi', 'Ivan' 'Role': ['Consultant', 'Consultant', 'Senior Associate', 'Consultant', 'Manager',

'Consultant', 'Senior Associate', 'Manager', 'Consultant', 'Consultant'], 'Location': ['Bangalore', 'Mumbai', 'Bangalore', 'Delhi', 'Mumbai',

'Mumbai', 'Delhi', 'Delhi', 'Bangalore', 'Delhi'],

'Experience': [2, 1, 5, 3, 7, 1, 6, 8, 4, 2],

'Compensation': [600000, 550000, 1200000, 650000, 1500000,

570000, 1250000, 1600000, 700000, 620000],

'Status': ['Active', 'Inactive', 'Active', 'Inactive', 'Active',

, , , , 'Rating': [3, 2, 4, 1, 5, 3, 2, 5, 1, 3]

}

df = pd.DataFrame(data) df.head()

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EmployeeID** | | **Name** | **Role** | **Location** | **Experience** | **Compensation** | **Status** | **Rating** |
| **0** 1 | | Alice | Consultant | Bangalore | 2 | 600000 | Active | 3 |
| **1** 2 | | Bob | Consultant | Mumbai | 1 | 550000 | Inactive | 2 |
| **2** | 3 | Charlie | Senior Associate | Bangalore | 5 | 1200000 | Active | 4 |
| **3** | 4 | David | Consultant | Delhi | 3 | 650000 | Inactive | 1 |
| **4** | 5 | Eve | Manager | Mumbai | 7 | 1500000 | Active | 5 |

print("Missing Values:\n", df.isnull().sum()) print("\nUnique Status Values:", df['Status'].unique()) print("\nSummary Stats:\n", df.describe())

Missing Values:

EmployeeID 0

Name 0

Role 0

Location 0

Experience 0

Compensation 0

Status 0

Rating 0

dtype: int64

Unique Status Values: ['Active' 'Inactive']

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Summary | Stats:  EmployeeID | Experience | Compensation | Rating |
| count | 10.00000 | 10.000000 | 1.000000e+01 | 10.000000 |
| mean | 5.50000 | 3.900000 | 9.240000e+05 | 2.900000 |
| std | 3.02765 | 2.514403 | 4.162051e+05 | 1.449138 |
| min | 1.00000 | 1.000000 | 5.500000e+05 | 1.000000 |
| 25% | 3.25000 | 2.000000 | 6.050000e+05 | 2.000000 |
| 50% | 5.50000 | 3.500000 | 6.750000e+05 | 3.000000 |
| 75% | 7.75000 | 5.750000 | 1.237500e+06 | 3.750000 |
| max | 10.00000 | 8.000000 | 1.600000e+06 | 5.000000 |

# Turnover by Role

turnover\_by\_role = df[df['Status'] == 'Inactive'].groupby('Role')['EmployeeID'].count().s print("Turnover by Role:\n", turnover\_by\_role)

# Turnover by Location

turnover\_by\_location = df[df['Status'] == 'Inactive'].groupby('Location')['EmployeeID'] c print("\nTurnover by Location:\n", turnover\_by\_location)

Turnover by Role:

Role

Consultant 3

Senior Associate 1

Name: EmployeeID, dtype: int64

Turnover by Location:

Location Delhi 2

Bangalore 1

Mumbai 1

Name: EmployeeID, dtype: int64

# Avg compensation by Role

print("\nAverage Compensation by Role:\n", df.groupby('Role')['Compensation'].mean())

# Avg compensation by Experience Buckets

df['ExperienceRange'] = pd.cut(df['Experience'], bins=[0, 2, 5, 10], labels=['0–2', '2–5' print("\nAverage Compensation by Experience:\n", df.groupby('ExperienceRange')['Compensat

Average Compensation by Role:

Role

|  |  |
| --- | --- |
| Consultant | 615000.0 |
| Manager | 1550000.0 |
| Senior Associate | 1225000.0 |
| Name: Compensation, | dtype: float64 |

Average Compensation by Experience:

ExperienceRange 0–2 585000.0

2–5 850000.0

5–10 1450000.0

Name: Compensation, dtype: float64

<ipython-input-9-f8eb0126108e>:6: FutureWarning: The default of observed=False is dep print("\nAverage Compensation by Experience:\n", df.groupby('ExperienceRange')['Com

# Check for patterns import seaborn as sns

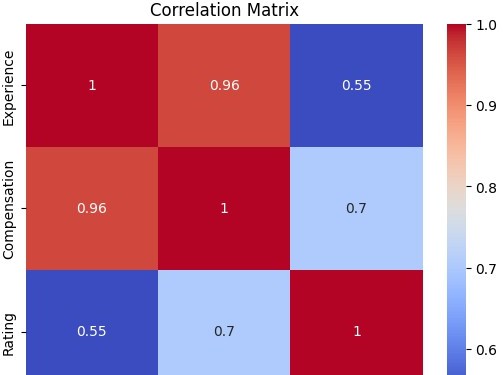
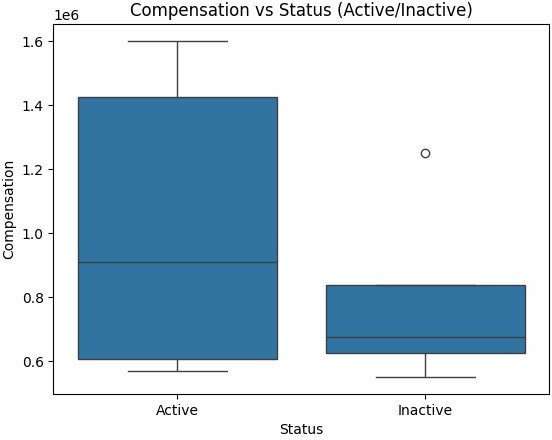
import matplotlib.pyplot as plt

sns.boxplot(data=df, x='Status', y='Compensation') plt.title("Compensation vs Status (Active/Inactive)") plt.show()

# Correlation heatma

corr = df[['Experience', 'Compensation', 'Rating']].corr() sns.heatmap(corr, annot=True, cmap="coolwarm") plt.title("Correlation Matrix")

plt.show()





# Filter senior roles

senior\_roles = df[df['Role'].isin(['Senior Associate', 'Manager'])]

# Bonus 10%

bonus\_impact = senior\_roles['Compensation'].sum() \* 0.10

# Stock grant of ₹200,000 per senior employee stock\_impact = len(senior\_roles) \* 200000

print(f"Total Bonus Payout (10%): ₹{bonus\_impact:,.0f}") print(f"Total Stock Grant Payout: ₹{stock\_impact:,.0f}")

Total Bonus Payout (10%): ₹555,000 Total Stock Grant Payout: ₹800,000

# Filter by role and status

def filter\_active\_by\_role(role):

filtered = df[(df['Role'] == role) & (df['Status'] == 'Active')] return filtered[['Name', 'Role', 'Location', 'Compensation']]

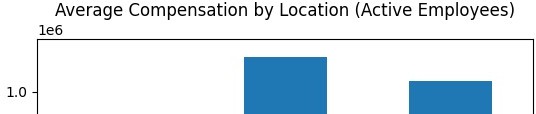
filter\_active\_by\_role("Consultant")

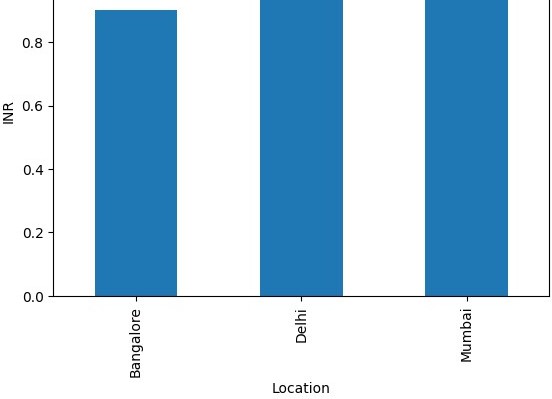
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Name** | **Role** | **Location** | **Compensation** |
| **0** | Alice | Consultant | Bangalore | 600000 |
| **5** | Frank | Consultant | Mumbai | 570000 |
| **9** | Judy | Consultant | Delhi | 620000 |

# Avg compensation by location (Bar Chart)

avg\_comp = df[df['Status'] == 'Active'].groupby('Location')['Compensation'].mean()

avg\_comp.plot(kind='bar', title="Average Compensation by Location (Active Employees)", yl plt.show()





experience\_group = df.groupby('ExperienceRange')['EmployeeID'].count() print("Employees by Experience Range:\n", experience\_group)

Employees by Experience Range:

ExperienceRange 0–2 4

2–5 3

5–10 3

Name: EmployeeID, dtype: int64

<ipython-input-15-dc4c84a6357a>:1: FutureWarning: The default of observed=False is de experience\_group = df.groupby('ExperienceRange')['EmployeeID'].count()

filtered\_data = df[['Name', 'Role', 'Location', 'Experience', 'Compensation', 'Status']] filtered\_data.to\_csv('Filtered\_Employee\_Data.csv', index=False)

print("CSV Exported Successfully")

CSV Exported Successfully

%%markdown

# Employee Compensation Forecasting App (TechSolve Inc.)

## Tools Used

* Python (Google Colab)
* pandas, seaborn, matplotlib

## Key Features

* Analyze headcount decline
* Compare compensation across roles/locations
* Simulate increments
* Export reports

## How to Run

1. Open in Google Colab
2. Upload data or use dummy
3. Execute each cell as per section

## Screenshots

\*Add visual outputs from charts here\*

**Employee Compensation Forecasting App (TechSolve Inc.)**

# Tools Used

 Python (Google Colab)

 pandas, seaborn, matplotlib

# Key Features

 Analyze headcount decline

 Compare compensation across roles/locations  Simulate increments

 Export reports

# How to Run

1. Open in Google Colab
2. Upload data or use dummy
3. Execute each cell as per section

# Screenshots

# Project Structure: EmployeeCompensationForecasting # Root Directory

# - data/

# - employees.csv # Raw employee data

# - location\_compensation.csv # Location-wise compensation reference #

# - notebooks/

# - 01\_Data\_Cleaning.ipynb

# - 02\_Filter\_Employees.ipynb

# - 03\_Compensation\_Analysis.ipynb

# - 04\_Increment\_Simulation.ipynb #

# - scripts/

# - create\_tables.py # Create SQLite tables # - filter\_employees.py

# - calculate\_average\_comp.py # - simulate\_increment.py

#

# - models/

# - employee.py

# - compensation.py #

# - utils/

# - db\_utils.py # DB helpers

# - viz\_utils.py # Visualization tools #

# - README.md

# - requirements.txt # List of Python packages # - .gitignore

# - LICENSE