India Job Market and Salary Trend

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", 120)
```

Load Dataset

In [3]:
```
df = pd.read_csv("C:/Users/ankit/OneDrive/Desktop/Projects/India job market and salary trend/Job_Market_India.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	Record_Date	Company_Name	Job_Role	Experience_Level	City	Salary_INR	Demand_Index	Remote_Option_Flag	Salary_Trend
0	2015-11-22 14:50:46.932655	State Bank of India	Product Manager	0-1 years	Noida	1615767	71	0	
1	2015-11-22 14:50:46.932655	Hindustan Unilever	Frontend Developer	0-1 years	Hyderabad	790163	81	0	
2	2015-11-22 14:50:46.932655	Bharat Petroleum	Software Engineer	5-8 years	Bangalore	1104756	94	0	
3	2015-11-22 14:50:46.932655	HDFC Bank	UI/UX Designer	5-8 years	Mumbai	905086	94	1	
4	2015-11-22 14:50:46.932655	Maruti Suzuki	Data Analyst	1-3 years	Bangalore	578687	35	0	

In [5]:
```
df.tail()
```

Out[5]:

	Record_Date	Company_Name	Job_Role	Experience_Level	City	Salary_INR	Demand_Index	Remote_Option_Flag	Salary_
29995	2025-11-18 14:50:46.932655	Aditya Birla Group	Digital Marketing Specialist	3-5 years	Pune	681751	23	1	
29996	2025-11-18 14:50:46.932655	Maruti Suzuki	Operations Manager	12+ years	Hyderabad	783833	38	0	
29997	2025-11-18 14:50:46.932655	Tata Consultancy Services	DevOps Engineer	8-12 years	Gurugram	2197581	78	0	
29998	2025-11-18 14:50:46.932655	HCL Technologies	Machine Learning Engineer	8-12 years	Hyderabad	2876168	38	0	
29999	2025-11-18 14:50:46.932655	Axis Bank	DevOps Engineer	5-8 years	Gurugram	1289682	93	1	

Size of data

In [6]:
```
print("Size of Data: " ,df.shape)
```

```
Size of Data:  (30000, 9)
```

Data Type

In [7]:
```
df.dtypes
```

Out[7]:
```
Record_Date            object
Company_Name           object
Job_Role               object
Experience_Level       object
City                   object
Salary_INR              int64
Demand_Index            int64
Remote_Option_Flag      int64
Salary_Trend_Pct      float64
dtype: object
```

Field Information

In [9]:
```
df.columns
```

Out[9]:
```
Index(['Record_Date', 'Company_Name', 'Job_Role', 'Experience_Level', 'City',
       'Salary_INR', 'Demand_Index', 'Remote_Option_Flag', 'Salary_Trend_Pct'],
      dtype='object')
```

Missing Values and Duplicates

In [10]:
```
print("Missing values per column:")
print(df.isnull().sum())
```

```
print("\nNumber of duplicate rows:", df.duplicated().sum())
```

```
Missing values per column:
Record_Date           0
Company_Name          0
Job_Role              0
Experience_Level      0
City                  0
Salary_INR            0
Demand_Index          0
Remote_Option_Flag    0
Salary_Trend_Pct      0
dtype: int64

Number of duplicate rows: 0
```

Convert Record_Date to datetime and sort

In [11]:
```
df['Record_Date'] = pd.to_datetime(df['Record_Date'], errors='coerce')
df = df.sort_values('Record_Date').reset_index(drop=True)
```

Descriptive statistics for numeric features

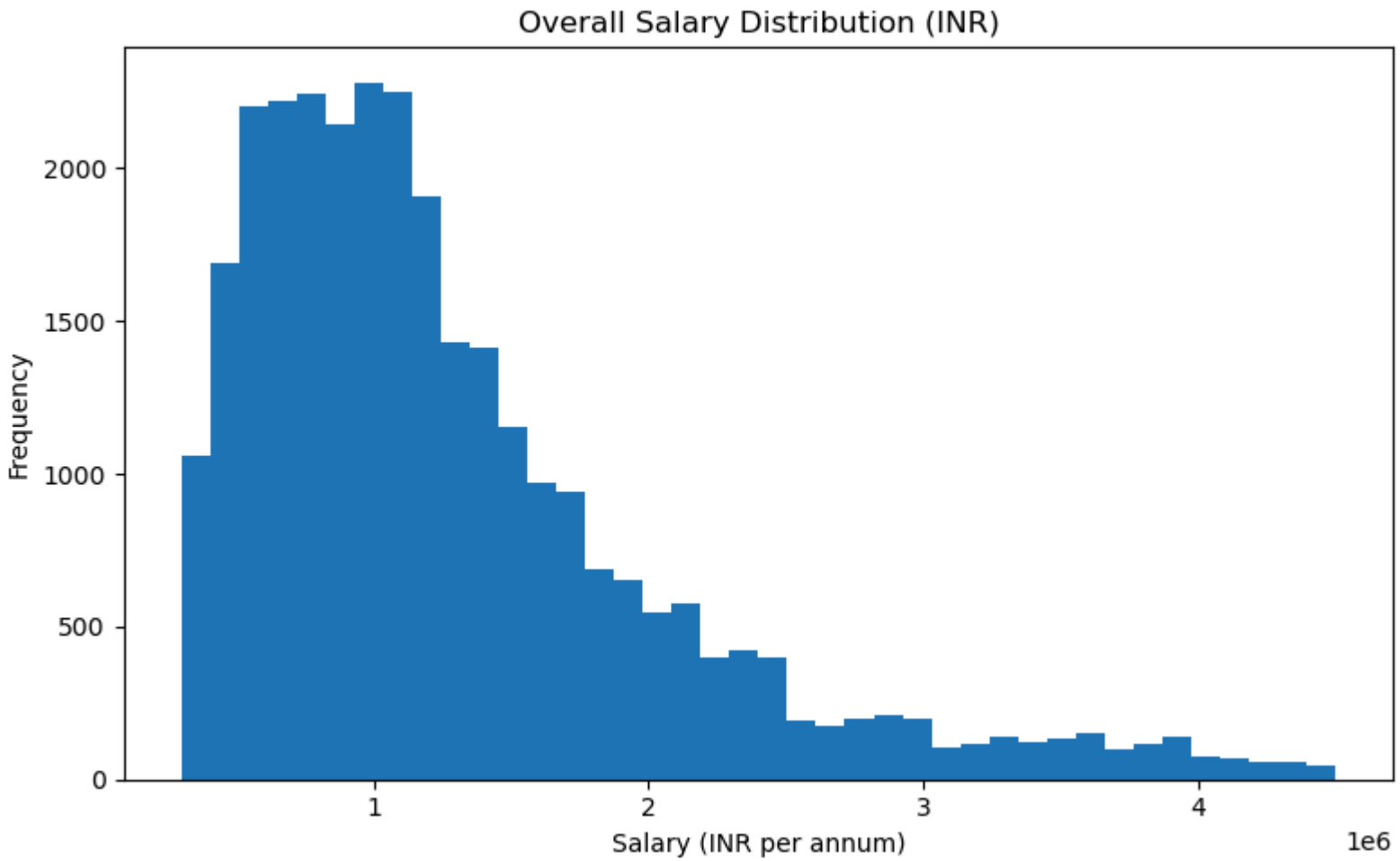
In [12]:
```
df.describe()
```

Out[12]:

	Record_Date	Salary_INR	Demand_Index	Remote_Option_Flag	Salary_Trend_Pct
count	30000	3.000000e+04	30000.000000	30000.000000	30000.000000
mean	2020-11-15 12:03:35.892655616	1.294258e+06	54.232600	0.501000	3.502035
min	2015-11-22 14:50:46.932655	3.000040e+05	10.000000	0.000000	-5.000000
25%	2018-05-28 14:50:46.932655104	7.350018e+05	32.000000	0.000000	-0.700000
50%	2020-11-11 14:50:46.932655104	1.088423e+06	54.000000	1.000000	3.450000
75%	2023-05-12 14:50:46.932655104	1.613194e+06	77.000000	1.000000	7.720000
max	2025-11-18 14:50:46.932655	4.499126e+06	99.000000	1.000000	12.000000
std	NaN	7.891613e+05	25.903033	0.500007	4.894683

Salary Distribution

In [13]:
```
plt.figure(figsize=(8, 5))
plt.hist(df['Salary_INR'], bins=40)
plt.title("Overall Salary Distribution (INR)")
plt.xlabel("Salary (INR per annum)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```



City-wise Salary Distribution

Choose top 10 cities by count

In [15]:
```
top_cities = df['City'].value_counts().head(10).index
subset = df[df['City'].isin(top_cities)]

plt.figure(figsize=(10, 5))
```

```
subset.boxplot(column='Salary_INR', by='City', rot=45)
plt.title("Salary by City")
plt.suptitle("")
plt.xlabel("City")
plt.ylabel("Salary (INR per annum)")
plt.tight_layout()
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```

Experience Level vs Salary

In [16]:
```
exp_order = ["0-1 years","1-3 years","3-5 years","5-8 years","8-12 years","12+ years"]
avg_salary_by_exp = df.groupby('Experience_Level')['Salary_INR'].mean().reindex(exp_order)

plt.figure(figsize=(8, 5))
plt.plot(avg_salary_by_exp.index, avg_salary_by_exp.values, marker='o')
plt.title("Average Salary by Experience Level")
plt.xlabel("Experience Level")
plt.ylabel("Average Salary (INR)")
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()

avg_salary_by_exp
```

Out[16]:
```
Experience_Level
0-1 years     1.302495e+06
1-3 years     1.301732e+06
3-5 years     1.288175e+06
5-8 years     1.296949e+06
8-12 years    1.276747e+06
12+ years     1.299456e+06
Name: Salary_INR, dtype: float64
```

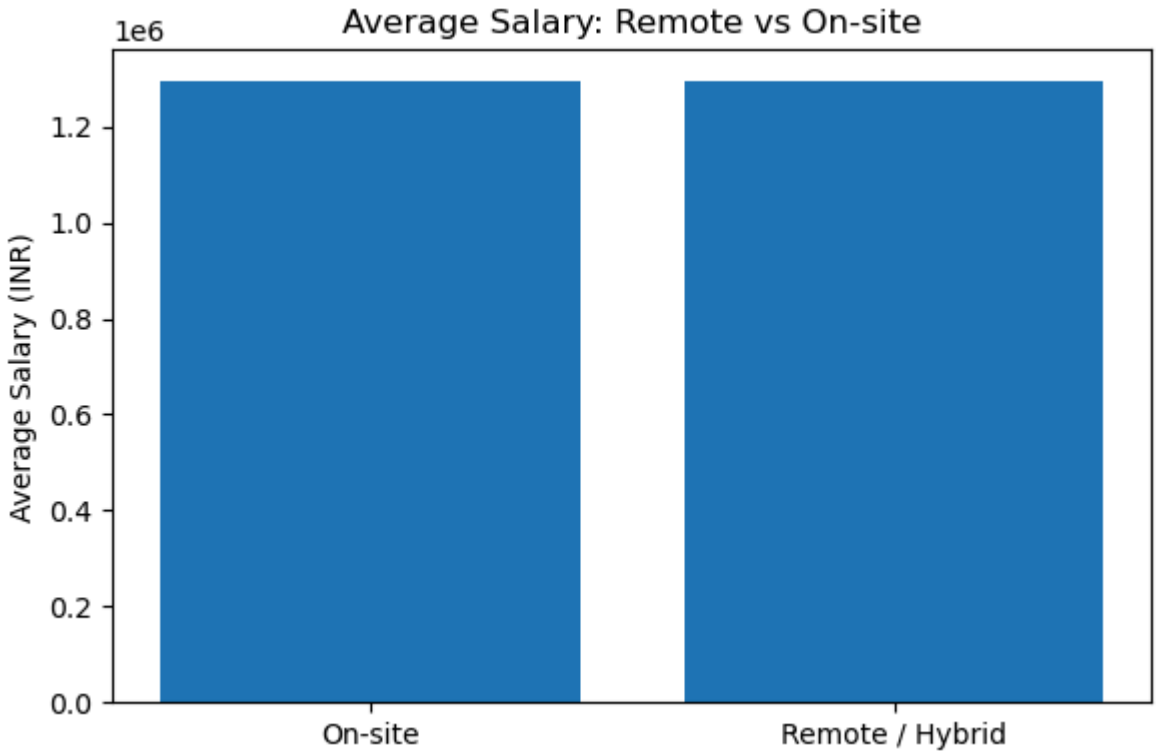
Remote vs On-Site Salaries

In [17]:
```
remote_map = {0: "On-site", 1: "Remote / Hybrid"}
df['Remote_Label'] = df['Remote_Option_Flag'].map(remote_map)

avg_salary_remote = df.groupby('Remote_Label')['Salary_INR'].mean()

plt.figure(figsize=(6, 4))
plt.bar(avg_salary_remote.index, avg_salary_remote.values)
plt.title("Average Salary: Remote vs On-site")
plt.ylabel("Average Salary (INR)")
plt.tight_layout()
plt.show()

avg_salary_remote
```



Out[17]:
```
Remote_Label
On-site            1.293475e+06
Remote / Hybrid    1.295038e+06
Name: Salary_INR, dtype: float64
```

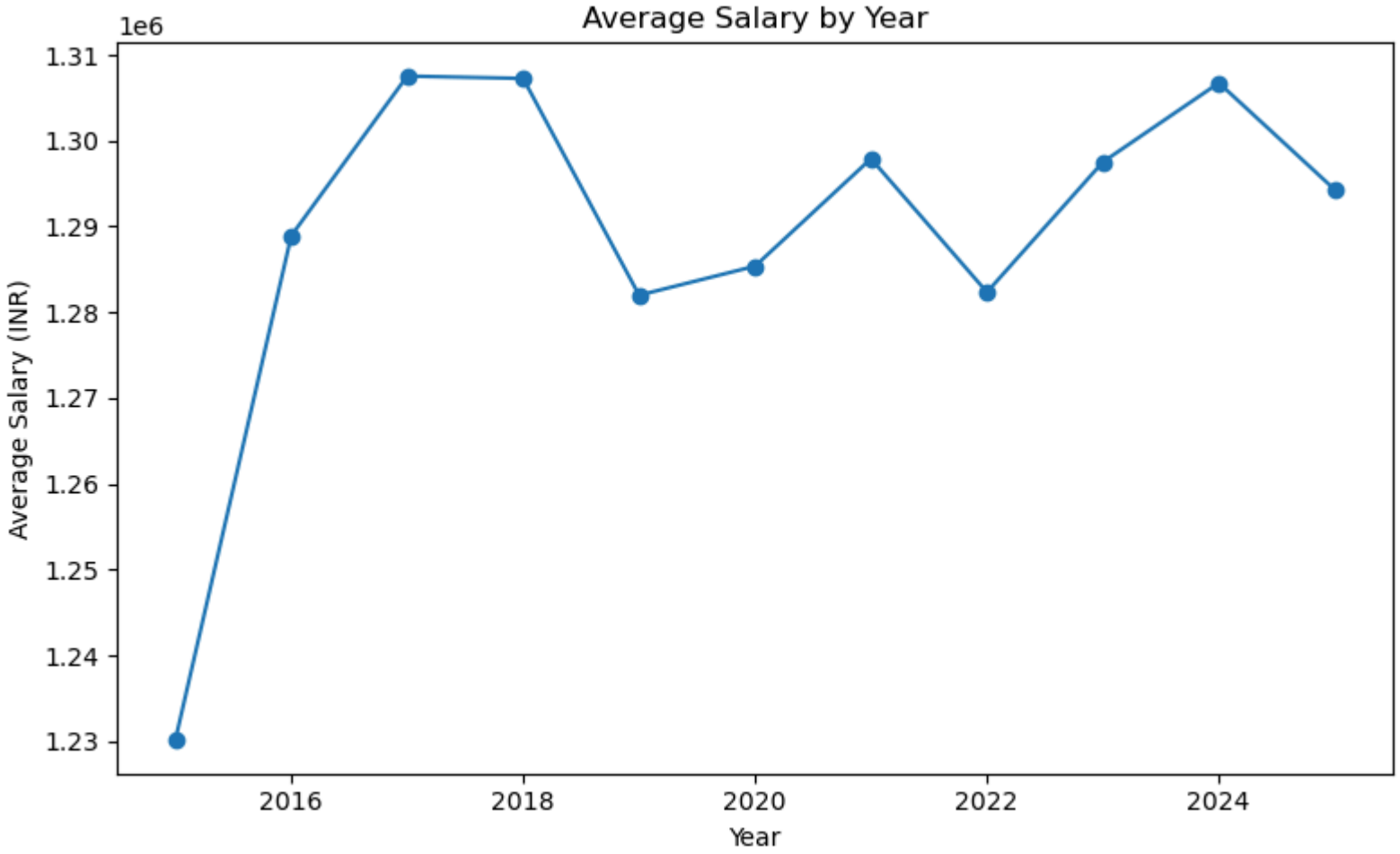
Time Trend: Average Salary Over Time

In [18]:
```
df['Year'] = df['Record_Date'].dt.year

avg_salary_year = df.groupby('Year')['Salary_INR'].mean()

plt.figure(figsize=(8, 5))
plt.plot(avg_salary_year.index, avg_salary_year.values, marker='o')
plt.title("Average Salary by Year")
plt.xlabel("Year")
plt.ylabel("Average Salary (INR)")
plt.tight_layout()
plt.show()

avg_salary_year
```


Average Salary by Year
1e6
1.31
1.30
1.29
1.28
1.27
1.26
1.25
1.24
1.23
Average Salary (INR)
2016
2018
2020
2022
2024
Year

Out[18]:
```
Year
2015    1.230128e+06
2016    1.288923e+06
2017    1.307535e+06
2018    1.307259e+06
2019    1.281995e+06
2020    1.285382e+06
2021    1.297908e+06
2022    1.282351e+06
2023    1.297520e+06
2024    1.306740e+06
2025    1.294290e+06
Name: Salary_INR, dtype: float64
```

Correlation Betweenn salary, demand, and trend metric.

In [19]:
```
numeric_cols = ['Salary_INR','Demand_Index','Remote_Option_Flag','Salary_Trend_Pct']
corr = df[numeric_cols].corr()

corr
```

Out[19]:

	Salary_INR	Demand_Index	Remote_Option_Flag	Salary_Trend_Pct
Salary_INR	1.000000	0.000949	0.000990	-0.003186
Demand_Index	0.000949	1.000000	0.008939	0.004887
Remote_Option_Flag	0.000990	0.008939	1.000000	-0.004152
Salary_Trend_Pct	-0.003186	0.004887	-0.004152	1.000000

Company-wise Salary Benchmarking

In [20]:
```
# Top 15 companies by job count
top_companies = df['Company_Name'].value_counts().head(15).index

company_salary = (
```

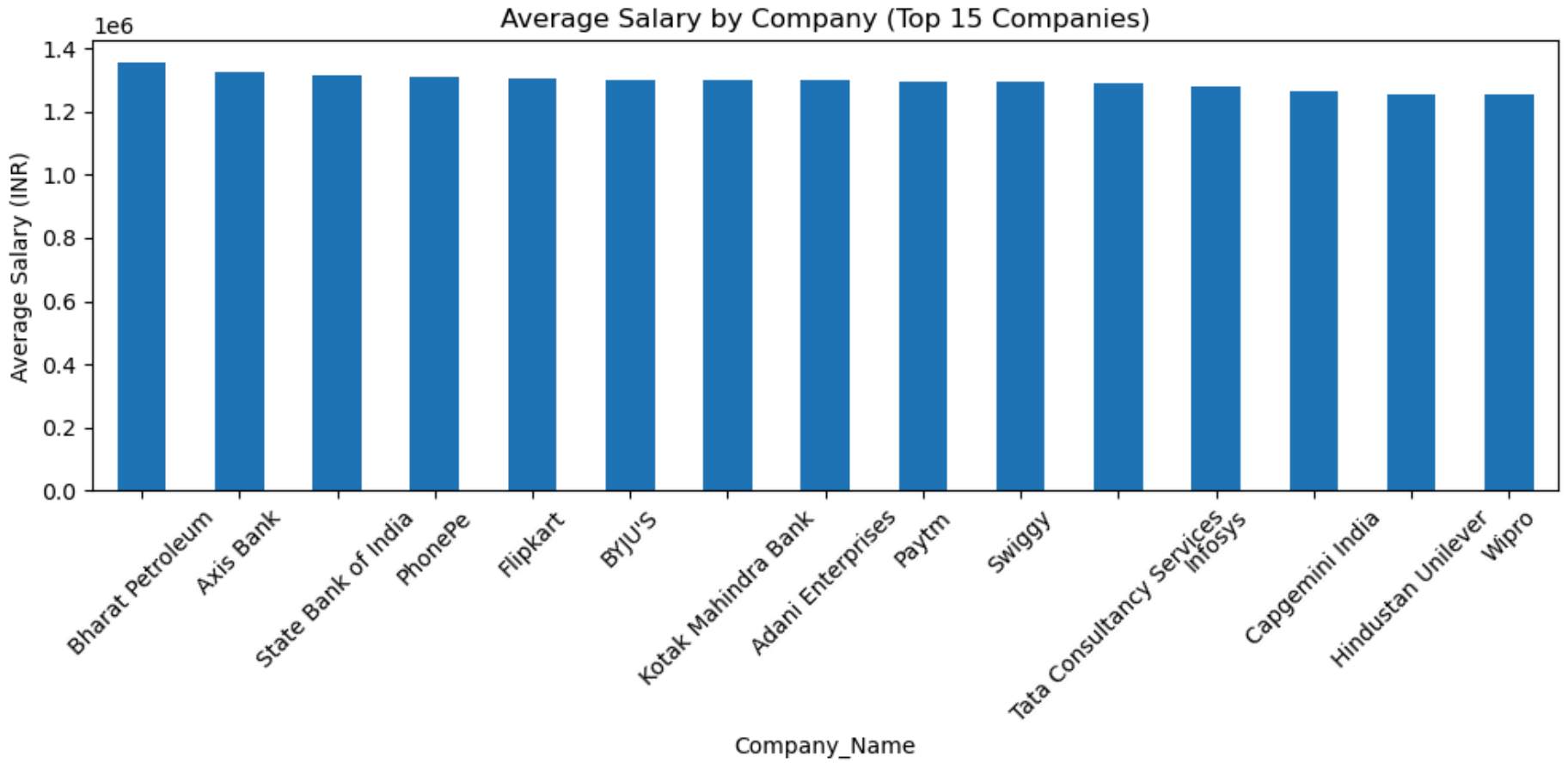
```
    df[df['Company_Name'].isin(top_companies)]
    .groupby('Company_Name')['Salary_INR']
    .agg(['mean', 'median', 'count'])
    .sort_values('mean', ascending=False)
)

company_salary
```

Out[20]:

Company_Name	mean	median	count
Bharat Petroleum	1.356502e+06	1117118.0	969
Axis Bank	1.325678e+06	1085387.0	945
State Bank of India	1.314763e+06	1097288.0	976
PhonePe	1.308899e+06	1132618.0	965
Flipkart	1.306978e+06	1095606.0	966
BYJU'S	1.300046e+06	1072146.0	954
Kotak Mahindra Bank	1.297975e+06	1092099.0	942
Adani Enterprises	1.297797e+06	1105300.0	1004
Paytm	1.294162e+06	1053233.5	998
Swiggy	1.292889e+06	1059849.5	942
Tata Consultancy Services	1.287206e+06	1075288.0	943
Infosys	1.278647e+06	1110021.0	970
Capgemini India	1.263606e+06	1041031.0	1002
Hindustan Unilever	1.253551e+06	1041386.0	978
Wipro	1.252828e+06	1074869.0	964

In [21]:
```
plt.figure(figsize=(10,5))
company_salary['mean'].plot(kind='bar')
plt.title("Average Salary by Company (Top 15 Companies)")
plt.ylabel("Average Salary (INR)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Role-Specific Salary Progression (Experience × Role)

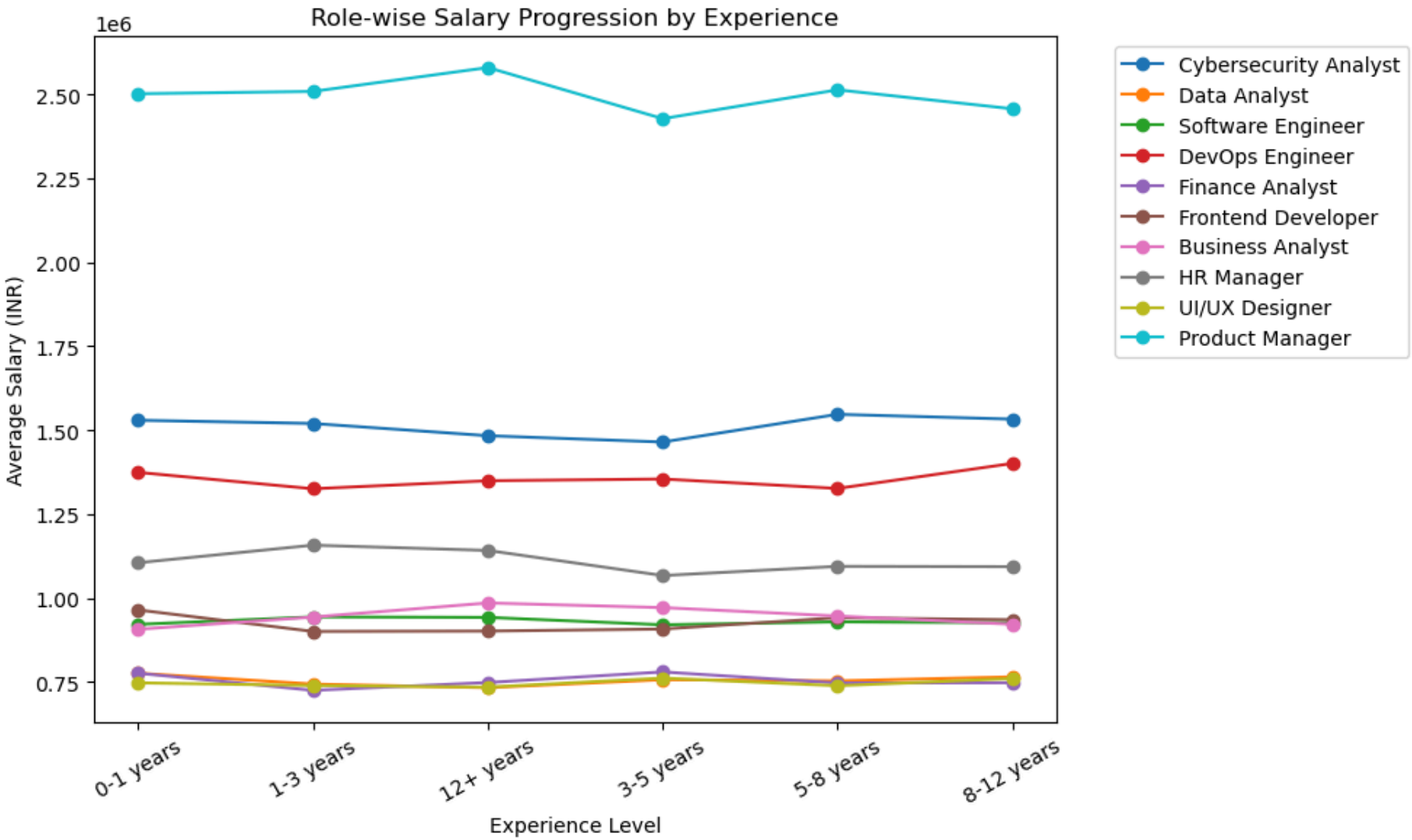
In [22]:
```
# Choose top 10 most common roles
top_roles = df['Job_Role'].value_counts().head(10).index

role_exp_salary = (
    df[df['Job_Role'].isin(top_roles)]
    .groupby(['Job_Role', 'Experience_Level'])['Salary_INR']
    .mean()
    .reset_index()
)
```

In [23]:
```
plt.figure(figsize=(10,6))

for role in top_roles:
    subset = role_exp_salary[role_exp_salary['Job_Role'] == role]
    plt.plot(subset['Experience_Level'], subset['Salary_INR'], marker='o', label=role)

plt.title("Role-wise Salary Progression by Experience")
plt.xlabel("Experience Level")
plt.ylabel("Average Salary (INR)")
plt.xticks(rotation=30)
plt.legend(bbox_to_anchor=(1.05,1), loc='upper left')
plt.tight_layout()
plt.show()
```



Demand Index vs Hiring Saturation Study

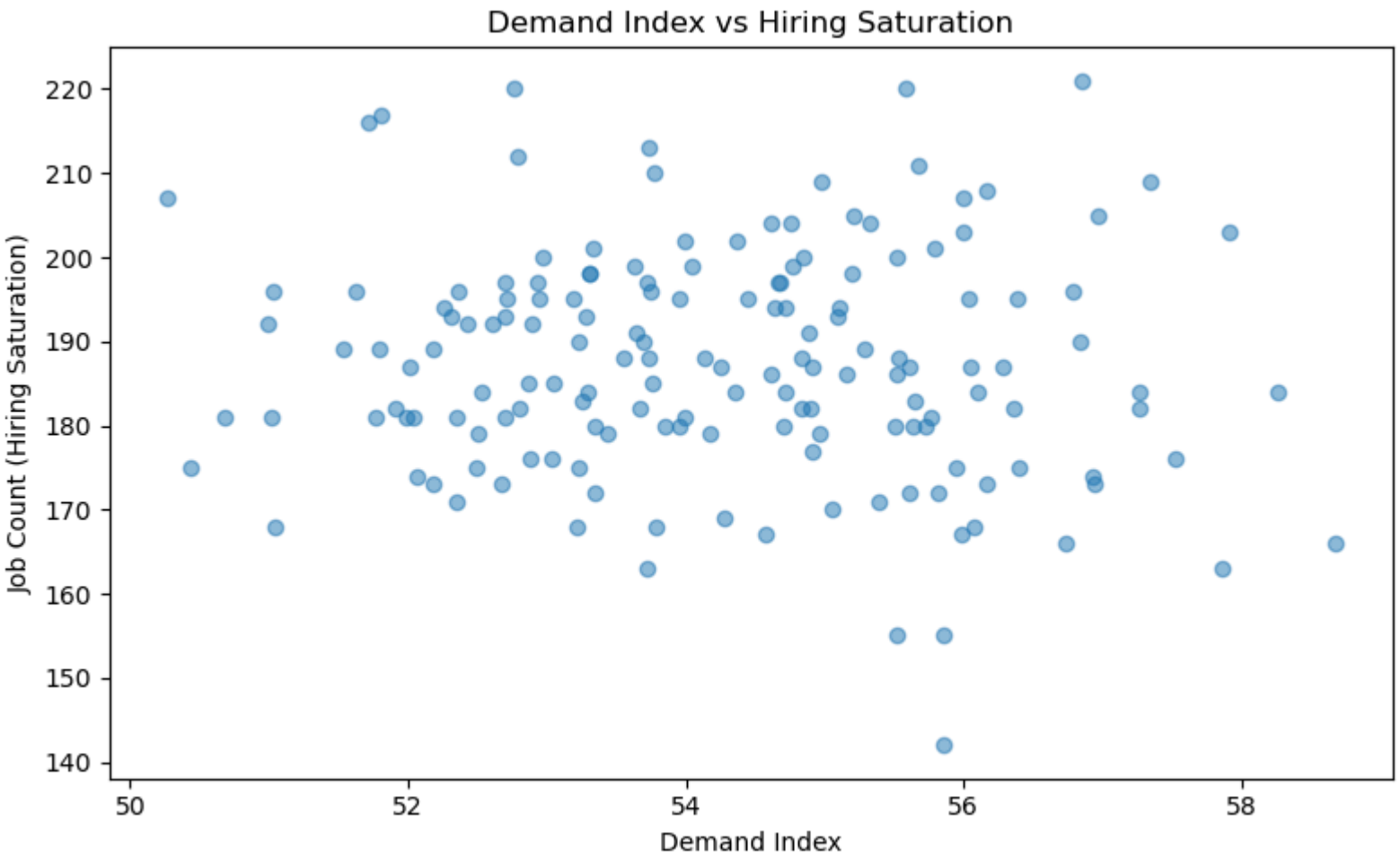
In [24]:
```
hiring_saturation = (
    df.groupby(['Job_Role', 'City'])
    .size()
    .reset_index(name='Job_Count')
```

```
)

demand_analysis = (
    df.groupby(['Job_Role', 'City'])['Demand_Index']
    .mean()
    .reset_index()
    .merge(hiring_saturation, on=['Job_Role', 'City'])
)
```

In [25]:
```
plt.figure(figsize=(8,5))
plt.scatter(
    demand_analysis['Demand_Index'],
    demand_analysis['Job_Count'],
    alpha=0.5
)

plt.title("Demand Index vs Hiring Saturation")
plt.xlabel("Demand Index")
plt.ylabel("Job Count (Hiring Saturation)")
plt.tight_layout()
plt.show()
```



Experience & Role Seniority

In [35]:
```
experience_map = {
    "0-1 years": 0.5,
    "1-3 years": 2,
    "3-5 years": 4,
    "5-8 years": 6.5,
    "8-12 years": 10,
    "12+ years": 15
```

```
}

df['Experience_Years'] = df['Experience_Level'].map(experience_map)

print(experience_map)
```

```
{'0-1 years': 0.5, '1-3 years': 2, '3-5 years': 4, '5-8 years': 6.5, '8-12 years': 10, '12+ years': 15}
```

In [36]:
```
senior_roles = [
    'Manager', 'Lead', 'Architect', 'Principal',
    'Director', 'Head'
]

df['Role_Seniority'] = df['Job_Role'].apply(
    lambda x: 1 if any(word in x for word in senior_roles) else 0
,
)

df[['Job_Role', 'Role_Seniority']].head(10)
```

Out[36]:

	Job_Role	Role_Seniority
0	Product Manager	1
1	Frontend Developer	0
2	Software Engineer	0
3	UI/UX Designer	0
4	Data Analyst	0
5	Digital Marketing Specialist	0
6	DevOps Engineer	0
7	Backend Developer	0
8	Data Scientist	0
9	Product Manager	1

ML Regression Model – Salary Prediction

In [28]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
```

In [29]:
```
features = [
    'Company_Name',
    'Job_Role',
    'City',
    'Experience_Years',
    'Demand_Index',
    'Remote_Option_Flag',
    'Role_Seniority'
]

X = df[features]
y = df['Salary_INR']
```

In [37]:
```
#categorical_features = ['Company_Name', 'Job_Role', 'City']

numeric_features = [
    'Experience_Years', 'Demand_Index',
    'Remote_Option_Flag', 'Role_Seniority'
]

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features),
        ('num', 'passthrough', numeric_features)
    ]
)

categorical_features = ['Company_Name', 'Job_Role', 'City']
numeric_features = [
```

```
    'Experience_Years', 'Demand_Index',
    'Remote_Option_Flag', 'Role_Seniority'
]

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features),
        ('num', 'passthrough', numeric_features)
    ]
)

print("Categorical:", categorical_features)
print("Numeric:", numeric_features)
```

```
Categorical: ['Company_Name', 'Job_Role', 'City']
Numeric: ['Experience_Years', 'Demand_Index', 'Remote_Option_Flag', 'Role_Seniority']
```

In [32]:
```
model = RandomForestRegressor(
    n_estimators=100,
    random_state=42,
    n_jobs=-1
)

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', model)
])
```

In [38]:
```
# Train & Evaluate

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

pipeline.fit(X_train, y_train)

y_pred = pipeline.predict(X_test)

print("R² Score:", r2_score(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
```

```
R² Score: 0.5262040764491965
MAE: 421267.5936883611
```

In [ ]: