# POLICY BAZAAR

## DATA ANALYST INTERVIEW QUESTIONS

## 0-3 YOE

## 10+ LPA

==========



# SQL

**Table: Customers**

This table stores information about the users who have registered on the Policybazaar platform.

| customer_id | customer_name | email | age | city | registration_date |
|---|---|---|---|---|---|
| C101 | Ankit Sharma | ankit.s@email.com | 35 | Delhi | 2023-01-15 |
| C102 | Priya Singh | priya.s@email.com | 28 | Mumbai | 2023-02-20 |
| C103 | Rahul Gupta | rahul.g@email.com | 45 | Bangalore | 2023-03-10 |
| C104 | Neha Mehta | neha.m@email.com | 32 | Mumbai | 2023-04-05 |
| C105 | Vishal Kumar | vishal.k@email.com | 50 | Chennai | 2023-05-18 |
| C106 | Ananya Roy | ananya.r@email.com | 29 | Delhi | 2023-06-25 |
| C107 | Karan Bhatia | karan.b@email.com | 41 | Hyderabad | 2024-01-08 |
| C108 | Sunita Rao | sunita.r@email.com | 38 | Bangalore | 2024-02-14 |
| C109 | Rajeev Verma | rajeev.v@email.com | 62 | Kolkata | 2024-03-22 |
| C110 | Pooja Jain | pooja.j@email.com | 25 | Mumbai | 2024-04-12 |

**Table: Policies**

This table contains details about the insurance policies available on the platform from different insurers.

| policy_id | policy_name | insurer | policy_type | sum_insured | premium_amount |
|---|---|---|---|---|---|
| P201 | Health Shield Plan | ICICI Lombard | Health | 500000 | 12000 |
| P202 | Life Secure Plus | HDFC Life | Life | 10000000 | 25000 |
| P203 | Car Protect Basic | Bajaj Allianz | Motor | 800000 | 8500 |
| P204 | Family Health Max | ICICI Lombard | Health | 1500000 | 30000 |
| P205 | Term Guard | HDFC Life | Life | 20000000 | 45000 |
| P206 | Travel Shield | Acko | Travel | 200000 | 5000 |
| P207 | Bike Protect | Bajaj Allianz | Motor | 150000 | 2500 |
| P208 | Senior Citizen Care | ICICI Lombard | Health | 1000000 | 20000 |

**Table: Transactions**

This table logs the sales of policies to customers.

| transaction_id | customer_id | policy_id | transaction_date | status |
|---|---|---|---|---|
| T301 | C101 | P201 | 2023-01-20 | Completed |
| T302 | C102 | P203 | 2023-02-25 | Completed |
| T303 | C103 | P202 | 2023-03-15 | Completed |
| T304 | C104 | P201 | 2023-04-10 | Pending |
| T305 | C101 | P205 | 2023-05-01 | Completed |
| T306 | C106 | P206 | 2023-06-30 | Completed |
| T307 | C107 | P204 | 2024-01-10 | Completed |
| T308 | C108 | P208 | 2024-02-15 | Completed |

| transaction_id | customer_id | policy_id | transaction_date | status |
|---|---|---|---|---|
| T309 | C109 | P202 | 2024-03-25 | Completed |
| T310 | C110 | P207 | 2024-04-15 | Completed |
| T311 | C104 | P201 | 2024-04-20 | Completed |
| T312 | C102 | P207 | 2024-04-22 | Pending |
| T313 | C101 | P202 | 2024-05-10 | Completed |

---

## Question 1

Find the top 3 insurers with the highest total premium_amount from all completed transactions.

```
SELECT
  p.insurer,
  SUM(p.premium_amount) AS total_premium
FROM Transactions AS t
JOIN Policies AS p
  ON t.policy_id = p.policy_id
WHERE
  t.status = 'Completed'
GROUP BY
  p.insurer
ORDER BY
  total_premium DESC
LIMIT 3;
```

---

## Question 2

Calculate the month-over-month growth rate of completed transactions. The growth rate should be displayed as a percentage.

```
WITH MonthlySales AS (
 SELECT
  DATE_FORMAT(transaction_date, '%Y-%m') AS sales_month,
  COUNT(transaction_id) AS total_sales
 FROM Transactions
 WHERE
  status = 'Completed'
 GROUP BY
  sales_month
)
SELECT
 sales_month,
 total_sales AS current_month_sales,
 LAG(total_sales, 1) OVER (
  ORDER BY
   sales_month
 ) AS previous_month_sales,
 ROUND(
  (
   (
    total_sales - LAG(total_sales, 1) OVER (
     ORDER BY
```

```
        sales_month
      )
    ) / LAG(total_sales, 1) OVER (
      ORDER BY
        sales_month
    )
  ) * 100,
  2
 ) AS growth_rate_percentage
FROM MonthlySales
ORDER BY
 sales_month;
```

---

## Question 3

Identify customers who have purchased more than one policy. Display their customer_name and the total number of policies they have purchased.

```
SELECT
 c.customer_name,
 COUNT(t.transaction_id) AS total_policies_purchased
FROM Customers AS c
JOIN Transactions AS t
 ON c.customer_id = t.customer_id
WHERE
 t.status = 'Completed'
```

```
GROUP BY
  c.customer_name
HAVING
  total_policies_purchased > 1
ORDER BY
  total_policies_purchased DESC;
```

---

## Question 4

Find the average premium_amount and sum_insured for each policy_type.

```
SELECT
  policy_type,
  AVG(premium_amount) AS avg_premium_amount,
  AVG(sum_insured) AS avg_sum_insured
FROM Policies
GROUP BY
  policy_type;
```

---

## Question 5

Create a report showing the customer_name, policy_name, insurer, and transaction_date for all transactions that occurred in the year 2024. Order the results by transaction_date.

```
SELECT
  c.customer_name,
```

```
  p.policy_name,
  p.insurer,
  t.transaction_date
FROM Transactions AS t
JOIN Customers AS c
  ON t.customer_id = c.customer_id
JOIN Policies AS p
  ON t.policy_id = p.policy_id
WHERE
  YEAR(t.transaction_date) = 2024
  AND t.status = 'Completed'
ORDER BY
  t.transaction_date;
```

---

## Question 6

Determine which city has the highest number of customers who have purchased a 'Health' policy.

```
SELECT
  c.city,
  COUNT(DISTINCT c.customer_id) AS number_of_health_policy_buyers
FROM Customers AS c
JOIN Transactions AS t
  ON c.customer_id = t.customer_id
JOIN Policies AS p
  ON t.policy_id = p.policy_id
```

```
WHERE
  p.policy_type = 'Health'
  AND t.status = 'Completed'
GROUP BY
  c.city
ORDER BY
  number_of_health_policy_buyers DESC
limit 1;
```

---

## Question 7

Find the customer who has the highest total premium_amount paid across all their completed policies. Display their customer_name and the total amount.

```
SELECT
  c.customer_name,
  SUM(p.premium_amount) AS total_premium_paid
FROM Customers AS c
JOIN Transactions AS t
  ON c.customer_id = t.customer_id
JOIN Policies AS p
  ON t.policy_id = p.policy_id
WHERE
  t.status = 'Completed'
GROUP BY
  c.customer_name
ORDER BY
```

```
 total_premium_paid DESC
LIMIT 1;
```

---

## Question 8

For each policy_type, find the oldest and youngest customers (by age) who have completed a transaction. Display the policy_type, customer_name, and age for both the oldest and youngest customers.

```
WITH CustomerRanks AS (
 SELECT
  p.policy_type,
  c.customer_name,
  c.age,
  RANK() OVER (
   PARTITION BY
    p.policy_type
   ORDER BY
    c.age ASC
  ) AS youngest_rank,
  RANK() OVER (
   PARTITION BY
    p.policy_type
   ORDER BY
    c.age DESC
  ) AS oldest_rank
```

```
 FROM Transactions AS t
 JOIN Customers AS c
  ON t.customer_id = c.customer_id
 JOIN Policies AS p
  ON t.policy_id = p.policy_id
 WHERE
  t.status = 'Completed'
)
SELECT
 policy_type,
 GROUP_CONCAT(
  CASE
   WHEN youngest_rank = 1
   THEN CONCAT(customer_name, ' (', age, ' years)')
   ELSE NULL
  END
 ) AS youngest_customer,
 GROUP_CONCAT(
  CASE
   WHEN oldest_rank = 1
   THEN CONCAT(customer_name, ' (', age, ' years)')
   ELSE NULL
  END
 ) AS oldest_customer
FROM CustomerRanks
WHERE
```

```
 youngest_rank = 1
 OR oldest_rank = 1
GROUP BY
 policy_type;
```

---

## Question 9

Calculate the percentage of 'Completed' transactions for each policy_type.

```
SELECT
 p.policy_type,
 SUM(
  CASE
   WHEN t.status = 'Completed'
   THEN 1
   ELSE 0
  END
 ) AS completed_transactions,
 COUNT(t.transaction_id) AS total_transactions,
 ROUND(
  SUM(
   CASE
    WHEN t.status = 'Completed'
    THEN 1
    ELSE 0
```

```
    END
   ) * 100 / COUNT(t.transaction_id),
   2
 ) AS completion_rate_percentage
FROM Transactions AS t
JOIN Policies AS p
  ON t.policy_id = p.policy_id
GROUP BY
  p.policy_type;
```

---

## Question 10

Identify customers who have pending transactions. For each such customer, list their customer_name and the policy_name of the pending policy.

```
SQL
SELECT
  c.customer_name,
  p.policy_name
FROM Customers AS c
JOIN Transactions AS t
  ON c.customer_id = t.customer_id
JOIN Policies AS p
  ON t.policy_id = p.policy_id
WHERE
```

```
 t.status = 'Pending';
```

==========

## PYTHON

SAMPLE DATAFRAME:

```
import pandas as pd

# Create the Customers DataFrame
customers_df = pd.DataFrame({
    'customer_id': ['C101', 'C102', 'C103', 'C104', 'C105', 'C106', 'C107', 'C108', 'C109', 'C110'],
    'customer_name': ['Ankit Sharma', 'Priya Singh', 'Rahul Gupta', 'Neha Mehta', 'Vishal Kumar', 'Ananya Roy', 'Karan Bhatia', 'Sunita Rao', 'Rajeev Verma', 'Pooja Jain'],
    'email': ['ankit.s@email.com', 'priya.s@email.com', 'rahul.g@email.com', 'neha.m@email.com', 'vishal.k@email.com', 'ananya.r@email.com', 'karan.b@email.com', 'sunita.r@email.com', 'rajeev.v@email.com', 'pooja.j@email.com'],
    'age': [35, 28, 45, 32, 50, 29, 41, 38, 62, 25],
    'city': ['Delhi', 'Mumbai', 'Bangalore', 'Mumbai', 'Chennai', 'Delhi', 'Hyderabad', 'Bangalore', 'Kolkata', 'Mumbai'],
    'registration_date': pd.to_datetime(['2023-01-15', '2023-02-20', '2023-03-10', '2023-04-05', '2023-05-18', '2023-06-25', '2024-01-08', '2024-02-14', '2024-03-22', '2024-04-12'])
})

# Create the Policies DataFrame
policies_df = pd.DataFrame({
    'policy_id': ['P201', 'P202', 'P203', 'P204', 'P205', 'P206', 'P207', 'P208'],
    'policy_name': ['Health Shield Plan', 'Life Secure Plus', 'Car Protect Basic', 'Family Health Max', 'Term Guard', 'Travel Shield', 'Bike Protect', 'Senior Citizen Care'],
```

```
    'insurer': ['ICICI Lombard', 'HDFC Life', 'Bajaj Allianz', 'ICICI Lombard', 'HDFC Life', 'Acko', 'Bajaj Allianz', 'ICICI Lombard'],
    'policy_type': ['Health', 'Life', 'Motor', 'Health', 'Life', 'Travel', 'Motor', 'Health'],
    'sum_insured': [500000, 10000000, 800000, 1500000, 20000000, 200000, 150000, 1000000],
    'premium_amount': [12000, 25000, 8500, 30000, 45000, 5000, 2500, 20000]
})

# Create the Transactions DataFrame
transactions_df = pd.DataFrame({
    'transaction_id': ['T301', 'T302', 'T303', 'T304', 'T305', 'T306', 'T307', 'T308', 'T309', 'T310', 'T311', 'T312', 'T313'],
    'customer_id': ['C101', 'C102', 'C103', 'C104', 'C101', 'C106', 'C107', 'C108', 'C109', 'C110', 'C104', 'C102', 'C101'],
    'policy_id': ['P201', 'P203', 'P202', 'P201', 'P205', 'P206', 'P204', 'P208', 'P202', 'P207', 'P201', 'P207', 'P202'],
    'transaction_date': pd.to_datetime(['2023-01-20', '2023-02-25', '2023-03-15', '2023-04-10', '2023-05-01', '2023-06-30', '2024-01-10', '2024-02-15', '2024-03-25', '2024-04-15', '2024-04-20', '2024-04-22', '2024-05-10']),
    'status': ['Completed', 'Completed', 'Completed', 'Pending', 'Completed', 'Completed', 'Completed', 'Completed', 'Completed', 'Completed', 'Completed', 'Pending', 'Completed']
})
```



---

## Question 1

Merge all three DataFrames to create a single master DataFrame for analysis. The final DataFrame should contain all information from the Customers, Policies, and Transactions tables.

```
# Merge Transactions with Policies
merged_df = pd.merge(transactions_df, policies_df, on='policy_id', how='left')

# Merge the result with Customers
master_df = pd.merge(merged_df, customers_df, on='customer_id', how='left')

# Display the first few rows of the master DataFrame
print("Master DataFrame:")
print(master_df.head())
```

---

## Question 2

Calculate the total premium_amount and sum_insured for each policy_type and display the results.

```
policy_summary = master_df.groupby('policy_type').agg(
   total_premium=('premium_amount', 'sum'),
   total_sum_insured=('sum_insured', 'sum')
)

print("\nSummary of Policies by Type:")
print(policy_summary)
```

---

## Question 3

Find the top 5 customers with the highest total premium_amount paid across all their completed policies. Display their names and the total amount.

```
top_customers = master_df[master_df['status'] == 'Completed'] \
   .groupby('customer_name')['premium_amount'].sum() \
   .nlargest(5)

print("\nTop 5 Customers by Total Premium Paid:")
print(top_customers)
```

---

## Question 4

Identify all customers who have purchased a 'Life' insurance policy. List their customer_name and city, ensuring each customer is listed only once.

```
life_policy_buyers = master_df[master_df['policy_type'] ==
'Life']['customer_name'].unique()
life_policy_df = master_df[master_df['policy_type'] == 'Life'] \
   [['customer_name', 'city']].drop_duplicates()

print("\nCustomers who purchased a 'Life' policy:")
print(life_policy_df)
```

---

## Question 5

Create a new column in the master DataFrame called age_group with the following categories:

- '18-30'
- '31-45'
- '46-60'
- '61+'

```
def categorize_age(age):
  if 18 <= age <= 30:
    return '18-30'
  elif 31 <= age <= 45:
    return '31-45'
  elif 46 <= age <= 60:
    return '46-60'
  else:
    return '61+'

master_df['age_group'] = master_df['age'].apply(categorize_age)

print("\nMaster DataFrame with 'age_group' column:")
print(master_df[['customer_name', 'age', 'age_group']].head())
```



---

## Question 6

Calculate the number of unique customers who registered and completed a transaction in the year 2024.

Python

```
# Customers who registered in 2024

registered_2024 = customers_df[customers_df['registration_date'].dt.year == 2024]


# Customers who had a completed transaction in 2024

completed_transactions_2024 = master_df[(master_df['transaction_date'].dt.year == 2024)
& (master_df['status'] == 'Completed')]


# Find the intersection of customer_id

registered_and_purchased_2024 = pd.merge(registered_2024,
completed_transactions_2024, on='customer_id')


num_customers = registered_and_purchased_2024['customer_id'].nunique()


print(f"\nNumber of unique customers who registered and completed a transaction in
2024: {num_customers}")
```

---



## Question 7

Calculate the average premium_amount and sum_insured for Health and Life policies purchased by customers in Mumbai.

```
mumbai_policies = master_df[(master_df['city'] == 'Mumbai') &
(master_df['policy_type'].isin(['Health', 'Life']))]


mumbai_summary = mumbai_policies.groupby('policy_type').agg(
```

```
    avg_premium=('premium_amount', 'mean'),
    avg_sum_insured=('sum_insured', 'mean')
)

print("\nAverage Premium and Sum Insured for Health/Life policies in Mumbai:")
print(mumbai_summary)
```

---

## Question 8

Find the insurer with the highest number of completed Motor policy transactions.

```
motor_insurer = master_df[(master_df['policy_type'] == 'Motor') & (master_df['status'] == 'Completed')] \
    .groupby('insurer')['transaction_id'].count() \
    .idxmax()

print(f"\nInsurer with the highest number of completed 'Motor' policy transactions: {motor_insurer}")
```

---

## Question 9

Create a pivot table that shows the total premium_amount for each insurer and policy_type.

```
pivot_table_premium = pd.pivot_table(
    master_df,
    values='premium_amount',
    index='insurer',
    columns='policy_type',
    aggfunc='sum',
    fill_value=0
)


print("\nPivot Table for Total Premium Amount by Insurer and Policy Type:")
print(pivot_table_premium)
```

---

## Question 10

Simulate the scenario of a new customer registration and a new policy purchase. Add a new row to both the customers_df and transactions_df to reflect this.

```
# New customer
new_customer = pd.DataFrame([{
    'customer_id': 'C111',
    'customer_name': 'New User',
    'email': 'new.user@email.com',
```

```
    'age': 30,
    'city': 'Pune',
    'registration_date': pd.to_datetime('2025-01-01')
}])
customers_df = pd.concat([customers_df, new_customer], ignore_index=True)

# New transaction for the new customer
new_transaction = pd.DataFrame([{
    'transaction_id': 'T314',
    'customer_id': 'C111',
    'policy_id': 'P201', # Let's say they buy the 'Health Shield Plan'
    'transaction_date': pd.to_datetime('2025-01-05'),
    'status': 'Completed'
}])
transactions_df = pd.concat([transactions_df, new_transaction], ignore_index=True)

print("\nUpdated Customers DataFrame (last row):")
print(customers_df.tail(1))
print("\nUpdated Transactions DataFrame (last row):")
print(transactions_df.tail(1))
```

---