Amazon Data Analytics Interview Questions
0-3 YOE
22-25 LPA

SQL Questions

---

1. Write a query to find the second-highest salary from an employee table.

Solution: To find the second-highest salary, we can use the DISTINCT clause along with the ORDER BY and LIMIT clauses.

Example:

```
SELECT MAX(Salary) AS Second_Highest_Salary

FROM Employees

WHERE Salary < (SELECT MAX(Salary) FROM Employees);
```

- The subquery (SELECT MAX(Salary) FROM Employees) finds the maximum salary.
- The outer query finds the maximum salary that is less than the highest salary.

---

2. How do you remove duplicates from a table in SQL?

Solution: To remove duplicates, we use the ROW_NUMBER() window function along with a DELETE statement.

Example:

```
WITH DuplicateCTE AS (
  SELECT *,
      ROW_NUMBER() OVER (PARTITION BY EmployeeID, Name, Department ORDER BY EmployeeID) AS Row_Num
  FROM Employees
)
DELETE FROM DuplicateCTE
WHERE Row_Num > 1;
```

- The ROW_NUMBER() assigns a unique number to duplicate rows.
- The DELETE operation removes rows where Row_Num is greater than 1.

---

3. Explain window functions and give an example.

Solution: Window functions perform calculations across a set of table rows related to the current row. Unlike aggregate functions, they do not collapse rows into a single result.

Example:

```
SELECT
  EmployeeID,
  Name,
  Salary,
  RANK() OVER (ORDER BY Salary DESC) AS Salary_Rank
FROM Employees;
```

- The RANK() function assigns a rank to each row based on salary in descending order.
- If two employees have the same salary, they will have the same rank, and the next rank will be skipped.

---

4. Write a query to find top N customers by total sales.

Solution: To find the top N customers by total sales, we can use the SUM() aggregate function along with ORDER BY and LIMIT.

Example:

```
SELECT CustomerID,
    SUM(SalesAmount) AS Total_Sales
FROM Sales
GROUP BY CustomerID
ORDER BY Total_Sales DESC
LIMIT 5;  -- Replace 5 with the desired N value
```

- The SUM(SalesAmount) calculates the total sales for each customer.
- The result is ordered in descending order, and the LIMIT clause restricts the result to the top N customers.

---

5. What is the difference between WHERE and HAVING?

Criteria	**WHERE**	**HAVING**
Application Level	Works on individual rows	Works on aggregated results
Aggregate Functions	Cannot be used	Can be used
Filtering Phase	Filters before aggregation	Filters after aggregation

Example:

```
-- Using WHERE
SELECT *
FROM Employees
WHERE Department = 'Sales';

-- Using HAVING
SELECT Department, AVG(Salary) AS Avg_Salary
```

```sql
FROM Employees

GROUP BY Department

HAVING AVG(Salary) > 50000;
```

- The WHERE clause filters rows before aggregation.
- The HAVING clause filters based on the aggregated result.

If you need further assistance or more questions solved, feel free to ask!

Data Analytics & Business Insights

---

6. How do you analyze customer churn using data?

Solution: Customer churn analysis identifies the reasons why customers stop doing business with a company. It helps develop strategies for customer retention.

Steps to Analyze Customer Churn:

1. **Data Collection:** Gather customer data like demographics, transaction history, usage frequency, complaints, feedback, etc.
2. **Define Churn:** Clearly define what constitutes churn, such as no transactions in the last 6 months.
3. **Feature Engineering:** Create features like average transaction value, days since the last transaction, tenure, etc.
4. **Exploratory Data Analysis (EDA):** Analyze patterns, distributions, correlations, and potential factors influencing churn.
5. **Predictive Modeling:** Apply classification techniques like Logistic Regression, Decision Trees, or Random Forest to predict churn.
6. **Customer Segmentation:** Segment customers based on demographics, behavior, and churn risk.
7. **KPIs:** Monitor churn rate, retention rate, customer lifetime value (CLV), etc.

Example Query (SQL):

```
SELECT
  CustomerID,
  COUNT(CASE WHEN TransactionDate >= DATEADD(MONTH, -6, GETDATE()) THEN 1 END) AS Transactions_Last_6_Months
FROM CustomerTransactions
GROUP BY CustomerID
HAVING COUNT(CASE WHEN TransactionDate >= DATEADD(MONTH, -6, GETDATE()) THEN 1 END) = 0;
```

This query identifies customers who have not made any transactions in the last 6 months.

---

7. How would you forecast sales for the next quarter?

Solution: Sales forecasting predicts future sales based on historical data. Accurate forecasts help in decision-making, inventory management, and marketing strategies.

Steps for Forecasting:

1. **Data Collection:** Gather historical sales data, seasonal trends, marketing campaigns, and economic factors.
2. **Data Preprocessing:** Handle missing values, remove outliers, and create time series data.
3. **Exploratory Data Analysis:** Analyze trends, seasonality, and cyclic patterns.
4. **Model Selection:** Use techniques like:
    - **Time Series Models:** ARIMA, SARIMA
    - **Machine Learning:** Random Forest, XGBoost
    - **Deep Learning:** LSTM (if the dataset is extensive)
5. **Evaluation:** Evaluate model performance using RMSE, MAE, and MAPE.
6. **Forecasting:** Predict sales for the next quarter and adjust strategies accordingly.

Example (Python with ARIMA):

```
from statsmodels.tsa.arima.model import ARIMA
```

```
# Assuming 'sales_data' is a DataFrame with 'Sales' column
model = ARIMA(sales_data['Sales'], order=(1, 1, 1))
result = model.fit()
forecast = result.forecast(steps=3)  # Forecasting for the next quarter (3 months)
print(forecast)
```

---

8. Explain A/B testing and how to interpret the results.

Solution: A/B testing compares two versions (A and B) of a variable (like a webpage) to determine which performs better.

Steps for A/B Testing:

1. **Define Objective:** Identify the metric to optimize, like click-through rate (CTR) or conversion rate.
2. **Formulate Hypothesis:** State the null hypothesis (no difference) and the alternative hypothesis (there is a difference).
3. **Sample Selection:** Randomly divide the audience into two groups — Group A (control) and Group B (test).
4. **Experiment Design:** Implement the changes in the test group while keeping the control group unchanged.
5. **Statistical Testing:** Use t-tests or chi-square tests to evaluate the significance.
6. **Analyze Results:** If the p-value < 0.05, reject the null hypothesis — the change has a significant impact.

Example:

```
SELECT
  Version,
  COUNT(*) AS Users,
  SUM(Conversion) AS Conversions,
  (SUM(Conversion) * 1.0 / COUNT(*)) AS Conversion_Rate
```

FROM AB_Testing_Data

GROUP BY Version;

- The query helps compare the conversion rates for versions A and B.

---

9. What metrics would you track for Amazon Prime subscription growth?

Solution: To track the growth of Amazon Prime subscriptions, monitor the following metrics:

- **Subscriber Growth Rate:** Percentage increase in subscriptions over time.
- **Churn Rate:** Percentage of subscribers who cancel their membership.
- **Retention Rate:** Percentage of users renewing their subscriptions.
- **Customer Lifetime Value (CLV):** Estimated revenue from a subscriber during their lifetime.
- **Average Revenue Per User (ARPU):** Revenue generated per subscriber.
- **Engagement Metrics:** Frequency of use, purchase behavior, video streaming hours, etc.
- **Conversion Rate:** Percentage of free-trial users converting to paid subscriptions.

Example Query (SQL):

```
SELECT
  COUNT(*) AS Total_Subscribers,
  SUM(CASE WHEN SubscriptionStatus = 'Cancelled' THEN 1 ELSE 0 END) AS Cancelled_Subscribers,
  (SUM(CASE WHEN SubscriptionStatus = 'Cancelled' THEN 1 ELSE 0 END) * 100.0 / COUNT(*)) AS Churn_Rate
FROM PrimeSubscriptions;
```

- This query calculates total subscribers and the churn rate.

---

10. How do you handle seasonality in sales data?

Solution: Seasonality refers to periodic fluctuations in data due to seasonal factors. Ignoring seasonality may lead to inaccurate forecasts.

Techniques to Handle Seasonality:

1. **Decomposition:** Separate time series into trend, seasonality, and residual components.
2. **Seasonal Adjustment:** Remove seasonality to analyze the underlying trend.
3. **Seasonal ARIMA (SARIMA):** Time series model that accounts for seasonality.
4. **Moving Averages:** Smooth out seasonal variations.
5. **One-Hot Encoding:** Represent seasons or months as categorical variables for ML models.

Example (Python for Seasonal Decomposition):

```
from statsmodels.tsa.seasonal import seasonal_decompose


result = seasonal_decompose(sales_data['Sales'], model='multiplicative', period=12)

result.plot();
```

- The decomposition shows the observed, trend, seasonal, and residual components.

Data Visualization (Power BI/Tableau)

---

11. How do you choose the right visualization for different datasets?

Solution: Choosing the right visualization depends on the purpose of data representation. Here's a guide:

- **Comparison:** To compare categories or groups
  - **Bar Chart, Column Chart:** Categorical comparisons

    - **Stacked Bar/Column Chart:** Breakdown of sub-categories
- **Distribution:** To understand the distribution of data
    - **Histogram:** Distribution of a single variable
    - **Box Plot:** Spread and outliers of data
- **Trend Analysis:** To track data changes over time
    - **Line Chart:** Continuous time-series data
    - **Area Chart:** Emphasizes magnitude over time
- **Part-to-Whole Relationship:** To analyze proportions
    - **Pie Chart, Donut Chart:** Share of total
    - **Stacked Bar/Area Chart:** Cumulative comparison
- **Correlation:** To see relationships between variables
    - **Scatter Plot:** Correlation and outliers
    - **Bubble Chart:** Relationship with an additional variable
- **Geographical Data:** Location-based insights
    - **Map Visualizations:** Regional and spatial data

Example:
In Power BI, use a **line chart** to analyze monthly sales trends and a **bar chart** to compare regional performance.

---

12. Explain drill-through & drill-down in Power BI.

Solution: Drill-through and drill-down help in analyzing data at different granularity levels.

- **Drill-Down:** Navigate from a summary level to a detailed view within the same visual.
    - Example: Click on a year in a bar chart to see monthly breakdowns.
    - **Implementation:** Enable drill-down in visuals and use hierarchical fields like Year ➔ Quarter ➔ Month.
- **Drill-Through:** Navigate from one page to another with a focus on specific data.

- Example: Right-click a region in a report to view detailed sales data on a separate page.
- **Implementation:** Create a new report page, add a "Drill-through" field, and set up "Back" navigation.

Example in Power BI:
To analyze regional sales:

1. Create a bar chart by **Region**.
2. Add a drill-through field like **Product Category** to view detailed sales by category.

---

13. What are parameters in Tableau, and how do you use them?

Solution: Parameters in Tableau are dynamic input values used for filtering, calculations, and control of visualizations.

Use Cases of Parameters:

- Filtering data dynamically (e.g., select top N customers)
- Switching between measures (e.g., Sales vs. Profit)
- Adjusting reference lines
- What-if analysis scenarios

Example:
Creating a parameter to switch between **Sales** and **Profit**:

1. **Create a Parameter:**
   - Name: Measure Selector
   - Data Type: String
   - Values: 'Sales', 'Profit'
2. **Create a Calculated Field:**

```
IF [Measure Selector] = 'Sales' THEN [Sales]
ELSE [Profit]
END
```

3. Use the calculated field in the visualization, and apply the parameter control.

---

14. How do you create a year-over-year growth visualization?

Solution: Year-over-year (YoY) growth compares metrics across different years to analyze growth trends.

In Power BI:

- Use **DAX** to create measures:

```
Total Sales = SUM(Sales[Amount])


YoY Growth =
CALCULATE(
   [Total Sales],
   SAMEPERIODLASTYEAR(Sales[Order Date])
)


YoY Growth % =
DIVIDE(
   [Total Sales] - [YoY Growth],
   [YoY Growth],
   0
)
```

- Create a **line and clustered column chart**:
    - X-axis: Year
    - Columns: Total Sales
    - Line values: YoY Growth %

In Tableau:

- Use **Table Calculations**:
    - Compute sales by year
    - Use a quick table calculation: **Year over Year Growth**
    - Adjust compute using "Specific Dimensions" for accurate analysis.

---

15. How do you optimize a Power BI dashboard for performance?

Solution: Optimizing a Power BI dashboard ensures faster load times and smoother user experiences.

Best Practices for Optimization:

1. **Data Modeling:**
    - Use a **Star Schema** instead of a Snowflake Schema.
    - Create relationships with integers instead of text.
2. **Efficient Measures and Calculations:**
    - Use **SUMX, AVERAGEX** judiciously.
    - Avoid calculated columns; prefer calculated measures.
3. **Reduce Data Load:**
    - Filter unnecessary data at the source level.
    - Use **DirectQuery** for large datasets if real-time data is needed.
4. **Query Optimization:**
    - Remove unused columns and tables.
    - Limit the number of visuals on a single report page.
5. **Visual Optimization:**
    - Disable **auto-date/time** for large datasets.
    - Reduce visuals using complex DAX measures.
    - Use **aggregations** for large fact tables.

Example of Optimized DAX:

Optimized Measure = CALCULATE([Total Sales], REMOVEFILTERS(Calendar[Year]))

- Here, REMOVEFILTERS improves performance by reducing filter context.

Statistics & Probability

16. What is normal distribution, and why is it important?

Solution:
A **normal distribution** is a symmetric, bell-shaped distribution of data where most of the observations cluster around the central peak (mean). It is defined by its **mean (μ)** and **standard deviation (σ)**.

Key Properties:

- Symmetrical around the mean
- Mean = Median = Mode
- 68% of data lies within ±1σ from the mean
- 95% of data lies within ±2σ from the mean
- 99.7% of data lies within ±3σ from the mean (Empirical Rule)

Importance:

- Many natural phenomena (height, weight, IQ scores) follow a normal distribution.
- It is the foundation for various statistical tests like **t-tests** and **confidence intervals**.
- Assumptions of normality are often necessary for parametric tests.

Example:
If the average IQ score is 100 with a standard deviation of 15:

- 68% of people have IQs between 85 and 115.

- 95% of people have IQs between 70 and 130.

---

17. Explain p-value in hypothesis testing.

Solution:
A **p-value** is the probability of obtaining a result at least as extreme as the observed result, assuming that the null hypothesis (**H_0**) is true.

- **Low p-value (< 0.05):** Strong evidence against **H_0** → Reject the null hypothesis.
- **High p-value (> 0.05):** Weak evidence against **H_0** → Fail to reject the null hypothesis.

Interpreting p-value:

- **$p < 0.01$:** Very strong evidence against **H_0**
- **$0.01 \le p < 0.05$:** Moderate evidence against **H_0**
- **$0.05 \le p < 0.1$:** Suggestive evidence, but not strong
- **$p \ge 0.1$:** Weak evidence

Example:
In an A/B test, suppose **$p = 0.03$** when comparing two marketing strategies.

- Since **$p < 0.05$**, there is significant evidence to reject the null hypothesis that both strategies perform the same.

---

18. How do you determine if a dataset is skewed?

Solution:
Skewness measures the asymmetry of the data distribution:

- **Right (Positive) Skew:** Tail extends to the right; Mean > Median > Mode
- **Left (Negative) Skew:** Tail extends to the left; Mean < Median < Mode
- **Zero Skew:** Symmetrical distribution

Methods to Determine Skewness:

- **Visualization:** Use histograms or box plots to observe skewness.
- **Skewness Coefficient:**

    - **Pearson's Skewness:** Skewness=3(Mean−Median) / Standard Deviation
    - If skewness > 0: Right skewed
    - If skewness < 0: Left skewed
- **Using Python (Pandas):**

```
import pandas as pd

data = pd.Series([5, 7, 9, 10, 15, 22, 30])

print(data.skew())  # Positive value indicates right skew
```

Example:
A dataset of house prices typically shows **right skew** because a few expensive houses inflate the mean.

---

19. What is the difference between correlation and causation?

Solution:

- **Correlation:** Measures the strength and direction of the relationship between two variables. It does not imply causation.
    - Ranges from -1 to +1.
    - **+1:** Perfect positive correlation
    - **0:** No correlation
    - **-1:** Perfect negative correlation
- **Causation:** Indicates that a change in one variable directly causes a change in another.
    - Requires experimental control to establish causality.
    - Not all correlated variables have a causal relationship.

Example:

- **Correlation:** Ice cream sales and drowning incidents are positively correlated. However, ice cream consumption does not cause drowning — it's due to summer temperatures.
- **Causation:** Increasing the number of study hours leads to better exam scores.

20. Explain the Central Limit Theorem (CLT) in simple terms.

Solution:
The **Central Limit Theorem** (CLT) states that the distribution of sample means approaches a normal distribution as the sample size increases, regardless of the original data distribution.

Key Points:

- Sample size ≥ 30 is generally considered sufficient.
- The mean of the sample means equals the population mean (μ).
- The standard deviation of sample means is called the **Standard Error (SE)**:
  $SE=\sigma/n$
  where **σ** is the population standard deviation and **n** is the sample size.

Importance:

- CLT allows us to apply normal distribution techniques to analyze sample means.
- Used in hypothesis testing and confidence interval estimation.

Example:
If we take the average height of random samples of 30 students from a population, the distribution of those sample means will approximate a normal distribution, regardless of the population's height distribution.

Python for Data Analytics

21. How do you handle missing values in Pandas?

Solution:
Handling missing values is crucial in data cleaning to ensure data quality. Pandas provides multiple ways to handle missing data.

Methods:

- **Identify Missing Values:**

```
import pandas as pd

df = pd.DataFrame({'Name': ['Alice', 'Bob', None], 'Age': [25, None, 30]})

print(df.isnull())     # Check missing values (True = Missing)

print(df.isnull().sum())  # Count of missing values per column
```

- **Drop Missing Values:**

```
df.dropna(inplace=True)  # Drops rows with any missing values
```

- **Fill Missing Values:**

```
df.fillna(0, inplace=True)  # Replace missing values with 0

df['Age'].fillna(df['Age'].mean(), inplace=True)  # Fill with mean
```

- **Forward/Backward Fill:**

```
df.fillna(method='ffill', inplace=True)  # Forward fill

df.fillna(method='bfill', inplace=True)  # Backward fill
```

Choosing the Right Method:

- Drop if missing data is minimal.
- Fill with statistical measures for numerical data.
- Forward/backward fill for time-series data.

---

22. Write a Python function to calculate the moving average of a time series.

Solution:
A **moving average** smoothens time-series data by averaging data points within a specific window.

```
import pandas as pd


def moving_average(data, window):
   return data.rolling(window=window).mean()
```

```
# Example

data = pd.Series([10, 20, 30, 40, 50, 60])

print(moving_average(data, window=3))
```

Output:

```
0    NaN
1    NaN
2   20.0
3   30.0
4   40.0
5   50.0
dtype: float64
```

- The first two values are NaN because the window size is 3.

Use Cases:

- Stock price trend analysis
- Smoothing noisy data in sensor readings

---

23. Explain groupby() and agg() in Pandas with an example.

Solution:

- **groupby():** Groups data based on one or more columns.
- **agg():** Applies aggregation functions like sum, mean, max, etc., on grouped data.

Example:

```
import pandas as pd

data = {'Department': ['HR', 'HR', 'IT', 'IT', 'Finance'],

     'Salary': [50000, 55000, 60000, 65000, 70000]}
```

df = pd.DataFrame(data)

grouped = df.groupby('Department').agg({'Salary': ['mean', 'sum']})

print(grouped)

Output:

```
          Salary
           mean    sum
Department
Finance      70000   70000
HR          52500   105000
IT          62500   125000
```

Explanation:

- **groupby('Department')** groups data by the 'Department' column.
- **agg({'Salary': ['mean', 'sum']})** applies mean and sum aggregations.

Use Cases:

- Summarizing sales data by region.
- Aggregating customer transactions by category.

---

24. How do you merge two datasets in Pandas?

Solution:
Merging in Pandas combines data from different DataFrames based on common columns or indexes. It is similar to SQL joins.

Merge Methods:

- **Inner Join:** Default; returns matching rows only.

import pandas as pd

df1 = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']})

df2 = pd.DataFrame({'ID': [2, 3, 4], 'Age': [25, 30, 35]})

merged_df = pd.merge(df1, df2, on='ID', how='inner')

print(merged_df)

Output:

```
  ID    Name  Age
0  2     Bob   25
1  3  Charlie   30
```

- **Left Join:** All from left, matching from right.

pd.merge(df1, df2, on='ID', how='left')

- **Right Join:** All from right, matching from left.

pd.merge(df1, df2, on='ID', how='right')

- **Outer Join:** All data from both, filling NaN for non-matches.

pd.merge(df1, df2, on='ID', how='outer')

Choosing the Right Merge:

- Use **inner** for common matches.
- Use **left** for retaining all left records.
- Use **outer** when retaining all data is crucial.

---

25. What is the difference between apply() and map() in Pandas?

Feature	**apply()**	**map()**
Purpose	Works on DataFrame or Series	Works only on Series
Scope	Applies functions element-wise	Maps values using a dictionary
Flexibility	Can apply complex functions	Typically for mapping/replacing

Feature	apply()	map()
Return Type	Series or DataFrame	Series

Examples:

- **apply() on DataFrame:**

```
import pandas as pd

df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})

df['Sum'] = df.apply(lambda row: row['A'] + row['B'], axis=1)

print(df)
```

Output:

```
  A  B  Sum
0  1  4    5
1  2  5    7
2  3  6    9
```

- **map() on Series:**

```
mapping = {1: 'One', 2: 'Two', 3: 'Three'}

df['Mapped_A'] = df['A'].map(mapping)

print(df)
```

Output:

```
  A  B  Sum Mapped_A
0  1  4    5      One
1  2  5    7      Two
2  3  6    9    Three
```

When to Use:

- Use **apply()** for complex operations.
- Use **map()** for simple value mappings.