Data Analyst Interview Questions (0-3 Years)
17-19 lpa

1. What is the difference between Primary Key and Foreign Key? (SQL Basics)

Primary Key uniquely identifies each row.
Foreign Key establishes relationships between two tables.

Example:

```
CREATE TABLE Department (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(50)
);

CREATE TABLE Employee (
    EmpID INT PRIMARY KEY,
    EmpName VARCHAR(50),
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES
Department(DeptID)
);
```

2. Write a query to find the second highest salary in the Employee table.

```
SELECT MAX(salary) AS SecondHighest
FROM Employee
WHERE salary < (SELECT MAX(salary) FROM
Employee
```

Explanation:

- Inner query finds the **highest salary**.
- Outer query finds the **maximum salary below that highest**, giving the **second highest**.

Tip:

Use DENSE_RANK() when multiple employees can have the same highest salary.

3. How do you handle missing values in a dataset? (Data Cleaning)

```
df['Age'].fillna(df['Age'].median(), inplace=True)
```

- Remove rows with too many missing values.
- Impute using mean/median/mode.
- Use forward fill/backward fill (time series).

- **Explanation:**
  Missing values can bias analysis. Choice depends on data context.
- **Tip:**
  Always check % of missing values before deciding the method.

4. What is the difference between COUNT(*), COUNT(column), and COUNT(DISTINCT column)?

COUNT(*) → counts all rows.
COUNT(column) → counts non-NULL values.
COUNT(DISTINCT column) → counts unique non-NULL values.

```
SELECT COUNT(*), COUNT(Salary),
COUNT(DISTINCT Salary)
FROM Employee;
```

Tip:
COUNT(*) is usually fastest for row counts.

5. What are measures of central tendency in statistics? (Stats Basics)

Mean (average)
Median (middle value)
Mode (most frequent value)

Example:
For salaries [40k, 45k, 50k, 200k] →

- Mean = 83.75k
- Median = 47.5k
- Mode = None

Tip:
Use Median when data has outliers.

6. What is a window function in SQL? Provide examples of ROW_NUMBER and RANK.

Definition:

A **window function** performs calculations **across a set of table rows** related to the current row — without collapsing rows like GROUP BY.

Syntax:

```
FUNCTION_NAME() OVER (PARTITION BY column ORDER BY column)
```

Example: ROW_NUMBER()

Assigns a unique sequential number to each row **within a partition**.

```
SELECT name, department, salary,
       ROW_NUMBER() OVER (PARTITION BY department ORDER BY salary DESC) AS row_num
FROM employees;
```

- Each employee within the same department gets a row number based on salary rank (highest first).

Example: RANK()

Assigns **the same rank** to rows with **equal values**, but skips the next rank(s).

```
SELECT name, department, salary,
       RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS rank_num
FROM employees;
```

- If 2 employees have the same salary, both get rank 1, and the next gets rank 3.

7. Write a query to fetch the top 3 performing products based on sales.

Assume table sales_data has:
product_id, product_name, total_sales

```
SELECT product_id, product_name, total_sales
FROM sales_data
ORDER BY total_sales DESC
LIMIT 3;
```

Alternate using RANK() (if ties matter):

```
SELECT product_id, product_name, total_sales
FROM (
  SELECT *, RANK() OVER (ORDER BY total_sales DESC) AS rank_num
  FROM sales_data
) ranked_sales
WHERE rank_num <= 3;
```

8. Explain the difference between UNION and UNION ALL.

Feature	UNION	UNION ALL
Duplicates	Removes duplicates	Keeps all rows, including duplicates
Performance	Slower (because of sorting)	Faster (no de-duplication)
Use case	When you want distinct rows	When duplicates are meaningful

Example:

```
SELECT city FROM customers
UNION
SELECT city FROM vendors;
```

→ Returns a unique list of cities.

```
SELECT city FROM customers
UNION ALL
SELECT city FROM vendors;
```

→ Returns **all cities**, including duplicates.

9. Explain p-value in hypothesis testing. (Statistics)

The p-value measures the probability of observing the data if the null hypothesis is true.

Example:

If p-value = 0.02 and α = 0.05 → Reject null hypothesis.

Tip:

Lower p-value = stronger evidence against null hypothesis.

10. How would you detect outliers in a dataset? (EDA)

Boxplot method (IQR rule: values < Q1–1.5*IQR or > Q3+1.5*IQR).
Z-score method (values with |z| > 3).

```
df[(df['Salary'] > df['Salary'].quantile(0.75) + 1.5*(df['Salary'].quantile(0.75)-df['Salary'].quantile(0.25)))]
```

11. Write a query to get the top 3 departments with the highest average salary. (SQL + Aggregation)

Syntax:

```
SELECT DeptID, AVG(Salary) AS AvgSal
FROM Employee
GROUP BY DeptID
ORDER BY AvgSal DESC
LIMIT 3;;
```

Benefits:

- Use DENSE_RANK() when ties exist.

12. What is correlation? How do you interpret it? (Statistics)

Correlation measures the linear relationship between two variables.

- r = 1 → Perfect positive.
- r = -1 → Perfect negative.
- r = 0 → No linear relationship.

:

Example:

- Height vs Weight → r = 0.85 (strong positive correlation).
- Correlation ≠ Causation..

13. Explain the difference between DELETE and TRUNCATE commands.

Feature	DELETE	TRUNCATE
Removes rows	Yes (can use WHERE condition)	Yes (removes all rows)
WHERE supported?	Yes	No
Logging	Logs each deleted row (slower)	Minimal logging (faster)
Rollback	Can be rolled back (if within transaction)	Can be rolled back (in some RDBMS)
Identity reset	Retains identity	Resets identity (in most DBs)
Use case	Partial deletion or audit trail needed	Full data wipe without audit needed

14. What are KPIs? Give examples for an e-commerce company. (Business)

KPIs = Key Performance Indicators to track business performance.
Examples for e-commerce:

- Conversion rate
- Average order value
- Customer acquisition cost
- Cart abandonment rate

15. How do you calculate a running total in SQL? (Window Functions – Advanced SQL)

Use SUM() with OVER (ORDER BY ...).

```
SELECT EmpID, Salary,
SUM(Salary) OVER (ORDER BY EmpID) AS RunningTotal
FROM Employee;
```

- Explanation:
  OVER defines a window.
  Rows are ordered, and running total is calculated cumulatively.

- Tip:
  Useful for cumulative sales, revenue, or tracking growth.

16. Explain the difference between Correlation and Regression. (Stats)

1. **Correlation → strength & direction of relationship between variables.**
2. **Regression → predicts one variable (Y) based on another (X).**

Example:

- Correlation: Height vs Weight (r = 0.85).
- Regression: Predict weight = 50 + 0.7*Height.

Tip:

Correlation is symmetric, Regression is asymmetric (Y depends on X).

17. How do you handle imbalanced datasets in classification problems? (ML + Analytics)

Oversampling minority (SMOTE).
Undersampling majority.
Use different metrics (Precision, Recall, F1).
Use cost-sensitive learning.

Tip:
Accuracy is misleading in imbalanced datasets — always check Recall/F1.

18. How would you design an A/B test for a new pricing model? (Experiment Design)

Answer

Define hypothesis (new price increases revenue).
Split users → Control (old price) vs Treatment (new price).
Random assignment to avoid bias.
Choose metric (Revenue/User, Conversion).
Run test → ensure significance ($p < 0.05$).

Tip:
Watch out for seasonality and sample bias.

19. How would you detect anomalies in financial transactions? (Real-World Case)

Answer:

- **Statistical:** Z-score, IQR method.
- **ML:** Isolation Forest, DBSCAN, Autoencoders.
- **Business rules:** Flag if transaction > 3x user's average.

Tip:
In practice, mix **rules + ML models** for anomaly detection.

Data Analysis/Scenario-Based Questions

20. Write a query to identify the most profitable regions based on transaction data.

Assume a transactions table:
(transaction_id, customer_id, amount, region, transaction_date)

Query to find top 3 profitable regions:

```
SELECT region, SUM(amount) AS total_revenue
FROM transactions
GROUP BY region
ORDER BY total_revenue DESC
LIMIT 3;
```

Explanation:

- Aggregates transaction amounts per region.
- Orders regions by total revenue.
- Retrieves top 3 using LIMIT.

Optional: You could also calculate profit by subtracting costs (if a cost column is present).

21. How would you analyze customer churn using SQL?

Step-by-step SQL approach:

Step 1: Define churn

Let’s say a churned customer is one who hasn’t transacted in the **last 6 months**.

Step 2: Sample schema

- customers(customer_id, name, signup_date)
- transactions(customer_id, transaction_date, amount)

Step 3: Query to identify churned customers

```
SELECT c.customer_id, c.name
FROM customers c
LEFT JOIN transactions t
  ON c.customer_id = t.customer_id
     AND t.transaction_date >= CURRENT_DATE - INTERVAL '6 months'
WHERE t.transaction_id IS NULL;
```

Step 4: Analyze churn metrics

You could extend this analysis by calculating:

- Churn rate = (Churned Customers / Total Customers) * 100
- Monthly churn trend
- Compare churned vs. active customers in terms of average spend

22. Explain the difference between OLAP and OLTP databases.

Feature	**OLTP (Online Transaction Processing)**	**OLAP (Online Analytical Processing**
Purpose	Handles real-time transactional queries	Used for analytical/reporting queries
Operations	INSERT, UPDATE, DELETE	SELECT (aggregate, group, slice, dice)
Data Structure	Highly normalized (3NF)	De-normalized (star/snowflake schema)
Speed	Fast for read/write of single rows	Fast for complex analytical queries
Examples	Banking systems, e-commerce order processing	Business intelligence, dashboards, sales trends
Users	Clerks, DBAs	Analysts, Data Scientists
Backup/Recovery	Essential and frequent	Less frequent

In short:

- **OLTP** = operational, fast, real-time transactions.
- **OLAP** = analytical, slow-changing, historical data.

23. How would you determine the Average Revenue Per User (ARPU) from transaction data?

ARPU = Total Revenue / Total Number of Users

Assume a transactions table:

(transaction_id, customer_id, amount, transaction_date)

SQL Query:

```
SELECT
  SUM(amount) * 1.0 / COUNT(DISTINCT customer_id) AS ARPU
```

```
FROM transactions;
```

Explanation:

- SUM(amount) gets total revenue.
- COUNT(DISTINCT customer_id) counts unique users.
- Multiply by 1.0 to ensure float division.

You can also compute monthly ARPU by grouping by month.

```
SELECT
  DATE_TRUNC('month', transaction_date) AS month,
  SUM(amount) * 1.0 / COUNT(DISTINCT customer_id) AS monthly_arpu
FROM transactions
GROUP BY month
ORDER BY month;
```

24. Describe a scenario where you would use a LEFT JOIN instead of an INNER JOIN.

Use LEFT JOIN when:

You want **all records from the left table**, even if there's **no matching record** in the right table.

Real-life Scenario:

Question: List all customers and their transactions — even if they haven't made any.

Query:

```
SELECT c.customer_id, c.name, t.transaction_id, t.amount
FROM customers c
LEFT JOIN transactions t
  ON c.customer_id = t.customer_id;
```

Why LEFT JOIN?

- Shows **all customers**, including those with **no transactions** (returns NULLs for those).
- Using INNER JOIN would exclude customers with zero activity.

25. Write a query to calculate YoY (Year-over-Year) growth for a set of transactions.

Assume a table named transactions with:
(customer_id, transaction_date, amount)

Step 1: Extract year-wise revenue

```
SELECT
  EXTRACT(YEAR FROM transaction_date) AS year,
  SUM(amount) AS total_revenue
FROM transactions
GROUP BY EXTRACT(YEAR FROM transaction_date);
```

Step 2: Calculate YoY Growth using a CTE and Self-Join

```
WITH yearly_revenue AS (
 SELECT
    EXTRACT(YEAR FROM transaction_date) AS year,
    SUM(amount) AS total_revenue
  FROM transactions
  GROUP BY EXTRACT(YEAR FROM transaction_date)
)
SELECT
  curr.year AS current_year,
  curr.total_revenue,
  prev.total_revenue AS previous_year_revenue,
  ROUND(((curr.total_revenue - prev.total_revenue) / prev.total_revenue) * 100, 2) AS
yoy_growth_percent
FROM yearly_revenue curr
LEFT JOIN yearly_revenue prev
  ON curr.year = prev.year + 1;
```

Explanation:

- Joins each year to its previous year.
- Computes YoY growth as a percentage.

26. How would you implement fraud detection using transactional data?

Fraud detection typically involves pattern recognition, anomaly detection, and rule-based filtering.

Possible SQL-Based Checks:

Type	**Rule**
Unusual Amounts	Flag transactions > 3x average amount of that user
Rapid Repeats	Detect multiple transactions from same user within seconds
Location Mismatch	Transactions from different countries within a short time
Card Sharing	Same card used by different customers or IPs

Example Query – Unusual high amount per user:

```
WITH avg_txn AS (
  SELECT customer_id, AVG(amount) AS avg_amount
  FROM transactions
  GROUP BY customer_id
)
SELECT t.*
FROM transactions t
JOIN avg_txn a
  ON t.customer_id = a.customer_id
WHERE t.amount > 3 * a.avg_amount;
```

27. Write a query to find customers who have used more than 2 credit cards for transactions in a given month.

Assume a transactions table:
(customer_id, card_id, transaction_date)

Query:

```
SELECT customer_id,
       TO_CHAR(transaction_date, 'YYYY-MM') AS txn_month,
       COUNT(DISTINCT card_id) AS cards_used
FROM transactions
GROUP BY customer_id, TO_CHAR(transaction_date, 'YYYY-MM')
HAVING COUNT(DISTINCT card_id) > 2;
```

Explanation:

- Groups by customer_id and month.
- Counts distinct card_id used.
- Filters where more than 2 cards were used in a month.

28. How would you approach a business problem where you need to analyze the spending patterns of premium customers?

Step-by-Step Structured Approach:

Step 1: Understand the Objective

- Clarify with stakeholders what **"spending pattern"** means.
  - o Is it frequency, amount, category, channel, or timing?
- Define **"premium customer"**:
  - o Based on credit score, card tier (e.g., Platinum, Centurion), monthly spend threshold, etc.

Step 2: Data Collection

- Gather relevant datasets:
  - o Customer table (ID, tier, demographics)
  - o Transactions table (amount, date, category, location)
  - o Cards table (card_type, limits, activation)

Step 3: Data Cleaning & Preparation

- Handle missing values and outliers.
- Filter only **premium customers** using defined criteria.
- Enrich data (e.g., categorize merchant types or locations).

Step 4: Exploratory Data Analysis (EDA)

Use SQL/Python/Power BI to derive insights like:

Focus Area	Example Analysis
Spend Amount	Average monthly/yearly spend
Time Trends	Seasonality or weekly spending behavior
Categories	Where they spend most (Travel, Dining, Shopping)
Geography	City or region-wise behavior

Focus Area	Example Analysis
Trends	Is their spend increasing/decreasing YoY?

Step 5: Segmentation

- Use clustering or thresholds to group premium customers into:
  - High spenders
  - Frequent spenders
  - Category loyalists (e.g., only travel)
- Identify anomalies or subgroups with unique patterns.

Step 6: Business Recommendations

- Personalize rewards or offers based on their dominant categories.
- Enhance retention strategies for segments showing decline.
- Promote premium card upgrades based on usage patterns.

Bonus: Sample SQL Query

Get top 3 spending categories of premium customers monthly:

```
SELECT customer_id,
       DATE_TRUNC('month', transaction_date) AS txn_month,
       category,
       SUM(amount) AS total_spend
FROM transactions
WHERE customer_id IN (
    SELECT customer_id FROM customers WHERE tier = 'Premium'
)
GROUP BY customer_id, txn_month, category
ORDER BY customer_id, txn_month, total_spend DESC;
```