COGNIZANT DATA ANALYST INTERVIEW QUESTIONS 0-3 YOE

---

1) How do you approach cleaning and preparing data for analysis?

Answer:

Cleaning and preparing data is a critical first step in the data analysis process, ensuring data is accurate, consistent, and usable. Here's how I typically approach it:

✅ **Step-by-Step Approach:**

1. **Understand the Dataset:**
    - Review column names, data types, and the number of unique values.
    - Use methods like .info(), .describe(), .head() in Python or schema review in SQL.
2. **Remove Duplicates:**
    - Check for and remove duplicate records using drop_duplicates() in pandas or DISTINCT in SQL.
3. **Handle Missing Values:**
    - Identify missing values using .isnull().sum() or COUNT(*) WHERE column IS NULL.
    - Handle them by:
        - **Dropping rows/columns** (if they have excessive missing values).
        - **Imputing** with mean, median, or mode.
        - **Forward/backward fill** for time-series data.

4. **Correct Data Types:**
    - Convert columns to appropriate types (e.g., dates to datetime, numeric strings to integers).
5. **Standardize & Normalize Data:**
    - Ensure consistent formatting (e.g., 'Yes'/'No' vs. 1/0).
    - Apply normalization or scaling techniques if required for ML models.
6. **Fix Inconsistencies:**
    - Fix inconsistent category labels like "Male", "M", "male" → "Male".
7. **Feature Engineering (Optional):**
    - Create new columns, such as extracting month from a date or computing total_price = quantity * price.
8. **Outlier Detection:**
    - Use statistical methods (e.g., IQR, Z-score) or visualizations (boxplots) to detect and handle outliers.

🧰 **Tools & Libraries Used:**

- **Python (pandas, numpy, scipy)**
- **SQL (Cleansing at database level)**
- **Excel (for initial review if needed)**

---

2) Explain how SQL is used in data analysis.

Answer:

SQL (Structured Query Language) is a fundamental tool in data analysis, especially when working with relational databases.

🔍 **Use of SQL in Data Analysis:**

1. **Data Extraction:**
    - SQL helps retrieve data from databases using SELECT statements.
2. **Data Filtering and Sorting:**

- Use WHERE, AND, OR, LIKE, BETWEEN, IN to filter data.
- Use ORDER BY to sort data for better visibility and understanding.

3. **Aggregations & Grouping:**
   - Perform summary statistics using COUNT(), SUM(), AVG(), MAX(), MIN().
   - Combine with GROUP BY for category-wise analysis.
4. **Joining Tables:**
   - Use INNER JOIN, LEFT JOIN, etc., to merge data from multiple tables.
   - Essential when working with normalized data spread across tables.
5. **Window Functions:**
   - Use ROW_NUMBER(), RANK(), LEAD(), LAG() for advanced calculations without collapsing rows.
6. **Data Cleaning in SQL:**
   - Handle null values with IS NULL, COALESCE().
   - Replace strings using REPLACE() or CASE WHEN statements.
7. **Subqueries and CTEs:**
   - Use nested queries and WITH statements for better readability and reusability.
8. **Data Validation and Checks:**
   - Check duplicates, data ranges, invalid entries directly in SQL before analysis.

✅ **Real-Life Scenario:**

Suppose you're working with a sales database — SQL helps in calculating:

```
SELECT region, SUM(sales) AS total_sales
FROM orders
WHERE order_date BETWEEN '2024-01-01' AND '2024-03-31'
GROUP BY region
ORDER BY total_sales DESC;
```

This gives regional sales trends, a typical analyst task.

---

3) How do you handle missing or incomplete data in a dataset?

Answer:

Handling missing or incomplete data is crucial to ensure the accuracy and robustness of your analysis.

🔍 **Approach to Handling Missing Data:**

1. **Identify Missing Values:**
    - In Python: df.isnull().sum()
    - In SQL: SELECT COUNT(*) FROM table WHERE column IS NULL
2. **Understand the Pattern:**
    - Determine if the data is missing at random (MAR), missing completely at random (MCAR), or not at random (MNAR).
3. **Common Techniques to Handle Missing Data:**

Technique	Use Case
Drop rows/columns	When missing values are very few and insignificant
Mean/Median Imputation	For numeric data, when distribution is symmetric/skewed
Mode Imputation	For categorical variables
Forward/Backward Fill	For time-series or sequential data
Predictive Imputation	Use ML models to predict missing values
Constant Value Imputation	Assign values like “Unknown” or 0

4. **Flag Missing Data (Optional):**
    - Create a new boolean column to mark missing entries, useful in ML models.
5. **Document Assumptions:**
    - Keep track of what and how you imputed or removed data for reproducibility.

🛠️ **Example in Python:**

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

# Fill categorical column with mode
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
```

🧠 **Important Consideration:**

Always assess the **impact** of missing data on your analysis before deciding on a strategy. Sometimes, missingness itself can be an insight (e.g., customers not responding to a question).

---

4) What is data normalization, and why is it important?

Answer:

🔍 **What is Data Normalization?**

Data normalization refers to the process of transforming data into a common scale **without distorting differences in the ranges of values**. It ensures that each feature contributes equally to the analysis or model.

It can also refer to a **database design technique** that organizes tables to reduce redundancy and improve data integrity.

✅ **Types of Normalization (in Data Preprocessing):**

1. **Min-Max Normalization:**
    - Scales values between 0 and 1.
    - Formula:

$$x' = x - \min(x) / (\max(x) - \min(x))$$

2. **Z-score Normalization (Standardization):**

- Scales data based on mean = 0 and standard deviation = 1.
- Formula:

$x' = (x - \mu)/\sigma$

3. **Decimal Scaling:**
    - Moves the decimal point of values to bring them within a standard range.

🎯 **Why is Normalization Important?**

- **Removes Bias Due to Scale:**
    - Algorithms like KNN, SVM, and Gradient Descent are distance-based and sensitive to scale.
- **Improves Model Accuracy:**
    - Balanced features lead to faster convergence and better results in machine learning.
- **Enhances Data Integrity (in Database Normalization):**
    - Prevents anomalies, ensures data is stored efficiently, and maintains relationships logically.

---

5) What tools and techniques do you use for data visualization?

Answer:

As a Data Analyst, data visualization is key to presenting findings in a clear, compelling way.

🧰 **Tools I Use for Visualization:**

Tool	Use Case
Power BI	Business dashboards, KPI reports, slicers, DAX-based visuals

Tool	Use Case
Tableau	Interactive dashboards, storyboards, drill-down charts
Excel	Quick analysis with pivot charts, bar/line/pie charts
Python (Matplotlib, Seaborn, Plotly)	Custom data visualizations and EDA plots
SQL (basic charting in some platforms)	Data summaries with export to Excel or BI tools

---

📊 **Common Visualization Techniques:**

Chart Type	Purpose
Bar/Column Chart	Comparing categories
Line Chart	Showing trends over time
Pie/Donut Chart	Showing proportions (use sparingly)
Histogram	Understanding distribution of numerical data
Boxplot	Identifying outliers and data spread
Heatmap	Visualizing correlation or matrix-based data
Scatter Plot	Finding relationships between two variables
TreeMap/Sunburst	Hierarchical data representation

💡 **Best Practices:**

- Choose the right chart for the right data.
- Avoid cluttered visuals.

- Use consistent color schemes.
- Highlight key insights with annotations.

---

6) Explain the difference between correlation and causation.

Answer:

🔗 **Correlation:**

- **Definition:** A **statistical relationship** or association between two variables.
- **Direction:** Can be **positive**, **negative**, or **zero**.
- **Measurement:** Pearson's correlation coefficient (r), ranging from -1 to 1.
- **Example:** Ice cream sales and temperature tend to rise together (positive correlation).

🔥 **Causation (Cause and Effect):**

- **Definition:** One variable **directly affects** the other.
- **Requires Evidence:** Correlation alone doesn't prove causation.
- **Example:** Smoking **causes** an increase in the risk of lung cancer.

⚖️ **Key Differences:**

Feature	Correlation	Causation
Meaning	Variables move together	One variable influences the other
Directionality	Not implied	Implied (cause → effect)
Proof Needed	Statistical relationship	Statistical + Experimental evidence
Example	Sleep hours ↔ Productivity	Exposure to virus → Illness

---

🚨 **Important Note:**

"Correlation does not imply causation."
Just because two variables move together doesn't mean one causes the other. Confounding variables or coincidence could be at play.

---

7) How do you perform hypothesis testing in data analysis?

Answer:

🍼 **Hypothesis Testing is a statistical method used to make inferences or draw conclusions about a population based on sample data.**

✅ **Step-by-Step Approach:**

1. **Formulate Hypotheses:**
    - **Null Hypothesis (H_0):** Assumes no effect or difference.
    - **Alternative Hypothesis (H_1):** Assumes there is an effect or difference.

Example: H_0: New marketing strategy has no effect on sales.
H_1: New marketing strategy increases sales.

2. **Choose the Right Test:**
    - Based on data type and distribution:
        - **Z-test/T-test:** Comparing means.
        - **Chi-square test:** Categorical variables.
        - **ANOVA:** Comparing means of 3+ groups.
        - **Proportion test:** For yes/no outcomes.
3. **Set the Significance Level (α):**
    - Common values: **0.05**, **0.01**.
    - Represents the risk of rejecting a true null hypothesis (Type I error).
4. **Calculate the Test Statistic and P-value:**
    - Use statistical formulas or tools (Python, Excel, R).

5. **Decision Making:**
    - If **p-value ≤ α**, reject H_0 → evidence supports H_1.
    - If **p-value > α**, fail to reject H_0.
6. **Interpret Results in Business Context:**
    - Clearly communicate findings to stakeholders.

🛠️ **Example in Python (T-test):**

```
from scipy.stats import ttest_ind

# Sales before and after campaign
before = [100, 120, 110, 115]
after = [130, 145, 140, 150]

t_stat, p_val = ttest_ind(before, after)

if p_val < 0.05:
    print("Reject H0 → Campaign has a significant effect")
else:
    print("Fail to reject H0 → No significant difference")
```


---

8) What is the difference between a primary key and a foreign key in databases?

Answer:

Both **Primary Key** and **Foreign Key** are used to establish and enforce relationships between tables in a **relational database**.

Feature	Primary Key	Foreign Key
Purpose	Uniquely identifies each record in a table	Links one table to another
Uniqueness	Must be **unique** and **non-null**	Can have **duplicates** and **nulls**
Location	Defined within the current table	Refers to a primary key in another table
Example	EmployeeID in Employees table	EmployeeID in Salaries referencing Employees

---

✅ **Real-Life Example:**

```
-- Primary Key
CREATE TABLE Employees (
  EmployeeID INT PRIMARY KEY,
  Name VARCHAR(50)
);

-- Foreign Key
CREATE TABLE Salaries (
  SalaryID INT PRIMARY KEY,
  EmployeeID INT,
  FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```


- EmployeeID is the **primary key** in Employees.
- The same EmployeeID acts as a **foreign key** in Salaries to maintain referential integrity.

---

9) How do you ensure data accuracy in your analysis?

Answer:

Ensuring **data accuracy** is essential for making reliable business decisions.

✅ **Steps I Follow:**

1. **Understand Data Source:**
    - Check data origin, collection method, and frequency.
2. **Validate Data Types & Formats:**
    - Ensure each column has the correct data type (e.g., date columns are in datetime format).
3. **Remove Duplicates and Inconsistencies:**
    - Use drop_duplicates() or SQL DISTINCT.
    - Standardize category values (e.g., "Male", "male", "M" → "Male").
4. **Handle Missing/Null Values:**
    - Identify missing data using .isnull() or IS NULL and impute appropriately.
5. **Cross-Check with Source Systems or Business Logic:**
    - Validate totals, summaries, or group-level data against raw system exports or reports.
6. **Outlier Detection:**
    - Use boxplots, IQR, or Z-scores to catch abnormal values.
7. **Use Checksums & Audit Logs (for large systems):**
    - Match row counts or hashes when importing/exporting data.
8. **Automated Testing:**
    - Use scripts or data validation tools to enforce rules (e.g., no negative sales values).
9. **Peer Reviews & Feedback:**
    - Discuss logic and findings with domain experts for verification.

🛠️ **Tools Used:**

- Python (pandas, NumPy, Pytest)
- SQL queries for integrity checks
- Power BI's data profiling feature
- Excel for quick cross-validation

---

10)

Write a query to find customers who have made more than one purchase on different dates and calculate their total purchase amount.

🟦 **Input Table: Sales**

sale_id	product_id	customer_id	sale_date	sale_amount
1	101	1001	2023-09-01	500
2	102	1002	2023-09-02	300
3	101	1001	2023-09-03	400
4	103	1003	2023-09-04	700
5	102	1002	2023-09-05	600

```
SELECT
  customer_id,
  COUNT(DISTINCT sale_date) AS distinct_purchase_days,
  SUM(sale_amount) AS total_purchase_amount
```

```
FROM
  Sales
GROUP BY
  customer_id
HAVING
  COUNT(DISTINCT sale_date) > 1;
```

✅ **Explanation:**

- COUNT(DISTINCT sale_date) → ensures we only count **unique dates** for purchases.
- HAVING COUNT(DISTINCT sale_date) > 1 → filters customers who purchased on **multiple dates**.
- SUM(sale_amount) → gives the **total amount spent** by each qualifying customer.

📜 **Result for your sample data:**

customer_id	distinct_purchase_days	total_purchase_amount
1001	2	900
1002	2	900

Both customers 1001 and 1002 purchased on two different dates.

---

11) There are two table Table A and Table B.

Table A

7
7
7

7

Table B

7

7

7

How many rows should be as output from below table queries?

A INNER JOIN B =

A LEFT JOIN B =

A UNION B =

A UNION ALL B =

A EXCEPT B =

A INTERSECT B =

A CROSS JOIN B WHERE A.ID=7 =

📘 **Table A:**

7

7

7

7 → 4 rows (value 7)

📘 **Table B:**

7

7

7 → 3 rows (value 7)

We're assuming there's an implicit ID column in each table with values = 7. Now let's analyze each SQL operation:

🔹 **A INNER JOIN B**

Each row from **A** joins with each matching row in **B** (on value = 7).

- A has 4 rows with 7
- B has 3 rows with 7

➡️ **Result = 4 × 3 = 12 rows**

🔹 **A LEFT JOIN B**

All 4 rows from A will join with matching rows in B (3 rows).

- For each of the 4 rows in A, there are 3 matches in B

➡️ **Result = 4 × 3 = 12 rows**

🔹 **A UNION B**

Removes duplicates (DISTINCT).

All values = 7

➡️ **Only one unique value = 7**

➡️ **Result = 1 row**

🔹 **A UNION ALL B**

Keeps all rows, including duplicates

- 4 from A
- 3 from B

➡️ **Result = 4 + 3 = 7 rows**

🔹 **A EXCEPT B**

Returns values in A that are **not in B**
All values in A are 7, and all values in B are 7

➡️ **Result = 0 rows**

🔹 **A INTERSECT B**

Returns common values (distinct) in A and B
Only value = 7 (in both)

➡️ **Result = 1 row**

🔹 **A CROSS JOIN B WHERE A.ID = 7**

Assuming CROSS JOIN is filtered with WHERE A.ID = 7 (and all values in A are 7), it's the same as:

SELECT *

FROM A

CROSS JOIN B

WHERE A.ID = 7;

All rows in A meet the condition (4 rows), and all 3 in B combine with each → 4 × 3 = **12 rows**

➡️ **Result = 12 rows**

✅ **Final Answers Recap:**

Query	Output Rows
A INNER JOIN B	12
A LEFT JOIN B	12
A UNION B	1

Query	Output Rows
A UNION ALL B	7
A EXCEPT B	0
A INTERSECT B	1
A CROSS JOIN B WHERE A.ID = 7	12