RAZORPAY DATA ANALYST INTERVIEW QUESTIONS (1ST ROUND)

CTC-22 LPA

SQL

---

1) How do you write a query to find duplicate rows in a table?

To find duplicate rows in a table, we need to identify records that have the same values in one or more columns.

Example Table: transactions

```
CREATE TABLE transactions (
    id INT PRIMARY KEY,
    user_id INT,
    amount DECIMAL(10,2),
    transaction_date DATE
);
```

Finding Duplicate Rows

Assume we want to find duplicate transactions based on user_id and amount.

```
SELECT user_id, amount, COUNT(*) AS duplicate_count
FROM transactions
GROUP BY user_id, amount
HAVING COUNT(*) > 1;
```

Explanation:

1. We use GROUP BY user_id, amount to group records that have the same user_id and amount.
2. COUNT(*) counts how many times each combination appears.
3. HAVING COUNT(*) > 1 filters only the duplicates.

If you need to retrieve full duplicate records including id, you can use:

```
SELECT *
FROM transactions
WHERE (user_id, amount) IN (
    SELECT user_id, amount
    FROM transactions
    GROUP BY user_id, amount
    HAVING COUNT(*) > 1
);
```

Prafik Jugant Mohapatra

2) How would you perform a LEFT JOIN and filter out NULLs in SQL?

Example Tables
We have two tables: customers and orders.

```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100)
);

CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_amount DECIMAL(10,2),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

LEFT JOIN with NULL Filter
We want to join both tables and filter out customers who have no orders.

```
SELECT c.customer_id, c.name, o.order_id, o.order_amount
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
WHERE o.order_id IS NOT NULL;
```

Explanation:

1. LEFT JOIN ensures all customers appear even if they don't have an order.
2. WHERE o.order_id IS NOT NULL removes rows where there are no matching records in the orders table.

If you want to find customers **without** orders (opposite case):

```
SELECT c.customer_id, c.name
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
WHERE o.order_id IS NULL;
```

---

3) What is a window function in SQL, and how do you use it for ranking data?

What is a Window Function?
A **window function** performs calculations across a set of table rows that are related to the

Ratik Jugant Mohapatra

current row within a defined **window (partition of data)**.

Ranking Data using Window Functions
Window functions used for ranking include:

- RANK()
- DENSE_RANK()
- ROW_NUMBER()

Example Table: sales

```
CREATE TABLE sales (
    sale_id INT PRIMARY KEY,
    salesperson VARCHAR(100),
    region VARCHAR(50),
    total_sales DECIMAL(10,2)
);
```

Using RANK()
Ranks salespeople based on total_sales (highest sales first).

```
SELECT salesperson, region, total_sales,
       RANK() OVER (PARTITION BY region ORDER BY total_sales DESC) AS rank
FROM sales;
```

Explanation:

1. PARTITION BY region creates separate ranking groups for each region.
2. ORDER BY total_sales DESC ranks highest sales as rank 1.
3. RANK() assigns the same rank to ties, skipping the next rank.

Difference Between Ranking Functions:

- RANK(): Skips ranks for ties.
- DENSE_RANK(): No gaps in ranking.
- ROW_NUMBER(): Assigns a unique row number.

Example:

```
SELECT salesperson, region, total_sales,
       RANK() OVER (ORDER BY total_sales DESC) AS rank,
       DENSE_RANK() OVER (ORDER BY total_sales DESC) AS dense_rank,
       ROW_NUMBER() OVER (ORDER BY total_sales DESC) AS row_num
FROM sales;
```

salesperson	total_sales	RANK()	DENSE_RANK()	ROW_NUMBER()
Alice	5000	1	1	1
Bob	5000	1	1	2
Charlie	4500	3	2	3
Dave	4000	4	3	4

Bratik Jugal Mohapatra

PYTHON

1.How do you import a CSV file into a pandas DataFrame and handle missing data?

Importing a CSV file into pandas

You can use the pd.read_csv() function to load a CSV file into a pandas DataFrame.

```
import pandas as pd

# Load CSV file into a DataFrame
df = pd.read_csv("data.csv")

# Display the first 5 rows
print(df.head())
```

Handling Missing Data

Once the CSV is loaded, you might encounter missing values. Here's how to handle them:

1. Detect missing values

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

2.Remove rows with missing values

```
df_cleaned = df.dropna()
```

This removes any row that has at least one missing value.

3.Fill missing values with a default value

```
df_filled = df.fillna(0)  # Replace NaN with 0
```

4. Fill missing values with the column mean/median/mode

```
df["salary"].fillna(df["salary"].mean(), inplace=True)  # Fill NaN in 'salary' column with mean value
df["department"].fillna(df["department"].mode()[0], inplace=True)  # Fill NaN in 'department' column with mode
```

5. Forward or Backward Fill

Pratik Jugant Mohapatra

```
df.fillna(method="ffill", inplace=True)  # Forward fill (propagate previous value)
df.fillna(method="bfill", inplace=True)  # Backward fill (propagate next value)
```

---

2) How do you use list comprehensions to filter and transform data?

List Comprehension Basics

List comprehensions provide a concise way to create lists in Python.

Example 1: Filter Even Numbers

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_numbers = [num for num in numbers if num % 2 == 0]
print(even_numbers)  # Output: [2, 4, 6, 8, 10]
```

Example 2: Apply Transformation

Convert a list of strings to uppercase.

```
names = ["alice", "bob", "charlie"]
uppercase_names = [name.upper() for name in names]
print(uppercase_names)  # Output: ['ALICE', 'BOB', 'CHARLIE']
```

Example 3: Filter and Transform Data in a List of Dictionaries

Let's assume we have employee data and want to extract names of employees earning more than 50,000.

```
employees = [
    {"name": "Alice", "salary": 60000},
    {"name": "Bob", "salary": 45000},
    {"name": "Charlie", "salary": 70000},
]

high_earners = [emp["name"] for emp in employees if emp["salary"] > 50000]
print(high_earners)  # Output: ['Alice', 'Charlie']
```

---

3) What are the differences between the apply() and map() functions in pandas?

Both apply() and map() are used to manipulate pandas Series and DataFrames, but they have key differences.

Function	Used On	Works On	Usage
map()	Series	Element-wise transformation	Works with functions, dictionaries, or Series

Pratik Ujgare Mohapatra

Function	Used On	Works On	Usage
apply()	Series & DataFrame	Row-wise or column-wise transformation	More flexible, can work on multiple columns

Example 1: Using map() on a Series

```
import pandas as pd

df = pd.DataFrame({
    "name": ["Alice", "Bob", "Charlie"],
    "salary": [60000, 45000, 70000]
})

# Applying a transformation to a single column
df["salary_in_lacs"] = df["salary"].map(lambda x: x / 100000)
print(df)
```

Output:

```
      name  salary  salary_in_lacs
0   Alice  60000            0.6
1     Bob  45000            0.45
2  Charlie  70000            0.7
```

Example 2: Using apply() on a DataFrame

Unlike map(), apply() works on both rows and columns.

Apply function to a column

```
df["updated_salary"] = df["salary"].apply(lambda x: x * 1.1)  # Increase salary by 10%
```

Apply function to multiple columns

```
df["name_length"] = df["name"].apply(len)  # Get length of each name
```

Apply function row-wise

```
df["category"] = df.apply(lambda row: "High" if row["salary"] > 50000 else "Low", axis=1)
```

Key Takeaway:

- Use map() for element-wise transformations on a **Series**.
- Use apply() for more complex row-wise or column-wise operations on a **DataFrame**.

EXCEL

1) How would you use VLOOKUP and XLOOKUP to merge data between two Excel sheets?

When working with multiple sheets in Excel, you can use **VLOOKUP** (for older versions) or **XLOOKUP** (for Excel 2019 & later) to fetch and merge data.

Prank Jugant Mohapatra

Using VLOOKUP
VLOOKUP helps to search for a value in the first column of a table and return a corresponding value from another column.
Example:
Imagine we have two sheets:

- **Sheet1** (Main Data)

Employee ID	Name	Department
101	Alex	HR
102	Bob	IT
103	Charlie	Sales

- **Sheet2** (Salary Data)

Employee ID	Salary
101	50,000
102	60,000
103	55,000

We want to fetch the **Salary** column from **Sheet2** into **Sheet1** using **VLOOKUP**.
VLOOKUP Formula:

=VLOOKUP(A2, Sheet2!A:B, 2, FALSE)

- A2 → The lookup value (Employee ID)
- Sheet2!A:B → The range where the lookup should occur
- 2 → The column number in the range (Salary is in column 2)
- FALSE → Exact match required

This will fetch salaries from **Sheet2** into **Sheet1**.

Using XLOOKUP
XLOOKUP is an improved version of VLOOKUP that allows searching both **left to right** and **right to left**, without needing a column number.
XLOOKUP Formula:

=XLOOKUP(A2, Sheet2!A:A, Sheet2!B:B)

- A2 → The lookup value (Employee ID)
- Sheet2!A:A → The column where the lookup should happen
- Sheet2!B:B → The column where the return value (Salary) is located

Advantages of XLOOKUP over VLOOKUP
✓ Works both left-to-right and right-to-left
✓ No need for a column index number
✓ Default exact match (no need for FALSE)
✓ Can return multiple values

---

2) What is the difference between absolute and relative cell references, and when would we use each?

Cell references in Excel determine how a formula behaves when copied to other cells. There

Pankaj Jugant Mohapatra

are three types:

Type	Example	Behavior
Relative	A1	Changes based on the new location when copied
Absolute	A1	Stays fixed regardless of where it's copied
Mixed	$A1 or A$1	Either row or column is fixed, but not both

Example 1: Relative Reference

=B2+C2

If copied from **D2** to **D3**, it will automatically change to:

=B3+C3

📌 **When to use:** When performing calculations across multiple rows/columns where values should adjust dynamically.

Example 2: Absolute Reference

=B2*D1

If copied down to other rows, B2 changes to B3, B4..., but D1 remains fixed.

📌 **When to use:** When referring to a constant value (e.g., tax rate, exchange rate) that shouldn't change.

Example 3: Mixed Reference

=B2*$D1

- $D1 → Column D is fixed, but row changes when copied down
- B$2 → Row 2 is fixed, but column changes when copied across

📌 **When to use:** When needing partial locking, like when copying formulas across multiple rows or columns.

---

3) How do you create a Pivot Table and analyze data with it?

Steps to Create a Pivot Table:

1. Select the dataset, e.g.:

Employee	Department	Sales	Region
Alex	HR	50,000	East
Bob	IT	60,000	West
Charlie	Sales	75,000	North
Diana	HR	40,000	East

2. Go to **Insert** → **PivotTable**
3. Choose **"New Worksheet"** or **"Existing Worksheet"**
4. Drag & Drop Fields into the PivotTable:
    - **Rows:** Department
    - **Values:** Sales (Sum)
    - **Columns (Optional):** Region
5. Click **OK**

Pivot Table Analysis

Example 1: Total Sales per Department

Department	Total Sales

Pratik Jugant Mohapatra

Department	Total Sales
HR	90,000
IT	60,000
Sales	75,000

Example 2: Sales per Department & Region

Department	East	West	North	Total
HR	90,000	0	0	90,000
IT	0	60,000	0	60,000
Sales	0	0	75,000	75,000

Pivot Table Features

✓ **Sort & Filter:** Click the dropdown in each field to sort or filter

✓ **Summarize Values:** Change **Sum** to **Average, Count, Max, Min**

✓ **Pivot Charts:** Go to **Insert** → **PivotChart** for visualization

📌 **Pivot tables are useful for quickly summarizing and analyzing large datasets!**

Final Takeaways

✓ Use **VLOOKUP/XLOOKUP** for merging data across sheets

✓ Understand **absolute vs. relative references** to build flexible formulas

✓ **Pivot tables** make data analysis easy with interactive reports

POWER BI

1) How would you create and customize a calculated column in Power BI?

What is a Calculated Column?

A **calculated column** is a column created in Power BI using **DAX (Data Analysis Expressions)**. It allows us to derive new information from existing columns.

Steps to Create a Calculated Column

1. Open **Power BI Desktop**.
2. Navigate to **Data View (Table Icon)**.
3. Select the table where you want to add the calculated column.
4. Click on **New Column** in the ribbon.
5. Enter a **DAX formula** to create the column.

Example 1: Creating a Full Name Column

Let's say we have a dataset with **First Name** and **Last Name** columns. We want to create a new column called **Full Name**.

DAX Formula:

```
Full Name = SalesData[First Name] & " " & SalesData[Last Name]
```

📌 This will concatenate the first and last name with a space in between.

Pratik Jugant Mohapatra

Example 2: Calculating Profit Margin
Assume we have the following columns in our **SalesData** table:

- Total Sales
- Total Cost

We can calculate **Profit Margin** as:

Profit Margin = (SalesData[Total Sales] - SalesData[Total Cost]) / SalesData[Total Sales]

📌 This formula helps in analyzing profitability.

Customization of Calculated Columns

- **Format Data Type**: Click on the column → Go to the "Modeling" tab → Change Data Type (Text, Number, Date, etc.).
- **Use Conditional Logic**: Example:

Discount Category = IF(SalesData[Total Sales] > 10000, "High Discount", "Low Discount")

📌 This categorizes sales into **High** and **Low Discount**.

---

2) What is the difference between a Slicer and a Filter in Power BI, and when would you use each?

What is a Slicer?
A **Slicer** is a **visual element** that allows users to interactively filter data in a report.
What is a Filter?
A **Filter** is a **report-level, page-level, or visual-level filter** that restricts the data displayed.

Feature	Slicer	Filter
Location	Visual (on the report)	In Filter Pane
Interactivity	Users can click to filter data	Mostly static (unless user edits)
Types	List, Dropdown, Date, Hierarchy, etc.	Basic, Advanced, Relative Date, Top N, etc.
Impact	Affects only the visuals linked to it	Can apply to the whole report, page, or a specific visual
Use Case	For interactive user-driven filtering	For predefined data control by the report designer

---

Example of Using a Slicer
Imagine we have a **Sales Report** and we want users to filter data based on **Region**.
Steps to Create a Slicer

1. Go to **Power BI Desktop** → **Report View**.
2. Click on **Slicer** in the Visualizations pane.
3. Drag the **Region** field into the slicer.
4. Now, users can click on a region (e.g., "East") to filter all visuals accordingly.

📌 **Use Case:** When we want users to dynamically filter the report by **Date, Category, or Location**.

Example of Using a Filter
If we want to **show only Sales above $50,000**, we can use a filter.
Steps to Apply a Filter

Pratik Jugan Mohapatra

1. Go to the **Filter Pane**.
2. Select the **Sales** visual.
3. Drag **Total Sales** to the **Visual-level filters**.
4. Set the filter condition:
    - **Show values where Total Sales > 50,000**.

📌 **Use Case:** When we want to exclude certain values (e.g., show only profitable regions, hide missing data, etc.).

---

3) How do you create relationships between tables in Power BI, and how do they impact your data model?

What is a Relationship in Power BI?
A **relationship** defines how tables are connected in Power BI. This allows us to analyze data from multiple tables together.

Steps to Create a Relationship

1. Go to **Power BI Desktop**.
2. Click on the **Model View** (⚡ icon).
3. Drag and drop fields to define relationships **OR** click **Manage Relationships**.
4. Choose the **Primary Key (PK)** from one table and the **Foreign Key (FK)** from another.
5. Set the **relationship type**:
    - **One-to-Many (1:M)** → Common in star schema models.
    - **Many-to-Many (M:M)** → Requires a bridging table.
    - **One-to-One (1:1)** → Rare but useful for unique mappings.
6. Click **Apply**.

Example: Relating a Sales Table and Customer Table
Sales Table

Order ID	Customer ID	Amount
101	C001	500
102	C002	700
103	C001	300

Customer Table

Customer ID	Customer Name	Region
C001	Alex	East
C002	Bob	West

To analyze **Total Sales by Customer**, we **link** Customer ID in both tables.

- **Primary Table:** Customer Table (Customer ID is unique)
- **Related Table:** Sales Table (Customer ID is repeated)

📌 **DAX Example to Calculate Total Sales by Customer**

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

Impact of Relationships in Power BI

1. **Better Data Organization:** Data remains normalized and avoids redundancy.
2. **Cross-table Analysis:** We can aggregate values across different tables.

Prajik Jugant Mohapatra

3. **Automatic Filtering:** If a **Customer Name** is selected, related sales filter automatically.
4. **Performance Optimization:** Queries run efficiently without excessive data duplication.

Final Takeaways

✓ **Calculated Columns** help in data transformation using **DAX expressions**.

✓ **Slicers vs. Filters** – Slicers are **interactive**, while Filters are **static** and report-wide.

✓ **Relationships** are essential for multi-table analysis, ensuring accurate **data connections**.

Pratik Jugant Mohapatra