Data Analyst Interview Questions

EXCEL

1. **VLOOKUP** – **Pull Data** from **Multiple Tables.**
2. **Pivot Table** – Sales Table – **Aggregate** Data with (**SUMIF/SUMIFS Function**) and **without using Formula**.
3. **Power Query** – **Split Data** (Alphanumeric) in Power Query – Example – AB1234567

SQL

1. Difference between **Delete, Drop, and Truncate**.
2. Difference between **Union and Union All**.
3. **Union Operation** on two **different structured** tables (Explained Below).
4. **Result of different Joins (Inner, Left, Right, Full, and Cross)** on Tables containing more than **1 Null Value** (Explain Below).
5. **Use of CASE**- Convert **Male to Female** and **Female to Male** by writing a single query.
6. Difference between **Rank, Dense Rank, and Row Number.**
7. Write a Query to **Find and Remove Duplicates** in the dataset.
8. Add Column (**Alter, Update**)- Create **Full Name from First Name and Last Name.**

Python

1. **Load a CSV file** into Python Environment – **Convert** it as a **Data Frame Object** – **Merge any two columns** of the Dataset – **Store the output as Excel**.
2. Write a **Function to get the Reverse Name** that should be able to **take input from the user** and show the reverse name as output.

Power BI

1. Difference between **SUM** and **SUMX**.
2. Explain the **Importance of Data Modelling.**
3. What is the **Star Schema and Snowflex schema**?
4. What is the difference between **a Fact and a Dimension Table**? How can we create a **Relationship between two Fact Tables**?

Behavioural Question

a. What is your thought about **Artificial Intelligence**? Don't you think it is going to replace us? (To see the attitude towards problems because AI is here to help and improve our efficiency not to replace.)

EXCEL

1. VLOOKUP

Given Dataset: *Three different tables will be provided, and the interviewee will be asked to fetch the **Name and SSN** of the customers **using a single formula only.***

Table A		
Id	**Name**	**SSN**
1	John Doe	123-45-6789
2	Jane Smith	987-65-4321
3	Michael Johnson	555-12-3456
4	Emily Brown	789-01-2345
5	Robert Lee	321-98-7654
6	Sarah Adams	456-78-9012
7	David Clark	234-56-7890
8	Lisa White	876-54-3210
9	James Taylor	789-12-3456
10	Mary Wilson	567-89-0123

Table B		
Id	**Name**	**SSN**
11	Daniel Hall	345-67-8901
12	Laura Miller	901-23-4567
13	Kevin Brown	678-90-1234
14	Amanda Harris	432-10-9876
15	Brian Adams	210-98-7654
16	Jessica Lee	543-21-6789
17	Christopher Smith	876-54-3210
18	Megan Johnson	123-45-6789
19	William Davis	987-65-4321
20	Olivia Taylor	555-12-3456

Table C		
Id	**Name**	**SSN**
21	Ethan Wilson	789-01-2345
22	Sophia Clark	321-98-7654
23	Benjamin Adams	456-78-9012
24	Ava Brown	234-56-7890
25	Noah White	876-54-3210
26	Grace Taylor	789-12-3456
27	Samuel Hall	567-89-0123
28	Natalie Miller	345-67-8901
29	Henry Brown	901-23-4567
30	Chloe Harris	678-90-1234

Formula:
=IFNA(VLOOKUP(A3,$E$2:$G$12,{2,3},0),IFNA(VLOOKUP(A3,$I$2:$K$12,{2,3},0),VLOOKUP(A3,$M$2:$O$12,{2,3},0)))

Question Set

Final Table		
Id	**Name**	**SSN**
4		
5		
7		
17		
19		
21		
24		
25		
28		

Solution Set

Final Table		
Id	**Name**	**SSN**
4	Emily Brown	789-01-2345
5	Robert Lee	321-98-7654
7	David Clark	234-56-7890
17	Christopher Smith	876-54-3210
19	William Davis	987-65-4321
21	Ethan Wilson	789-01-2345
24	Ava Brown	234-56-7890
25	Noah White	876-54-3210
28	Natalie Miller	345-67-8901

2. PIVOT TABLE and SUMIF Function

Given Data Set: *Amount Column is of **Text Type**. Need to **convert to Numeric Data Type First**.*
To show the aggregated total amount of each Customer, one can take any of the below approaches.

1. ***Formula-Based Approach:** Use of **SumIF/SUMIFS** function.*
2. ***Non-Formula-Based Approach:** Use of **Pivot Table**.*

Date of Purchase	Customer Name	Amount ($)
2024-01-05	John Smith	150
2024-01-10	Mary Johnson	200
2024-01-15	David Lee	180
2024-01-20	Emily Chen	220
2024-01-25	Michael Brown	170
2024-02-05	John Smith	160
2024-02-10	Mary Johnson	210
2024-02-15	David Lee	190
2024-02-20	Emily Chen	230
2024-02-25	Michael Brown	180
2024-03-05	John Smith	170
2024-03-10	Mary Johnson	220
2024-03-15	David Lee	200
2024-03-20	Emily Chen	240
2024-03-25	Michael Brown	190
2024-04-05	John Smith	180
2024-04-10	Mary Johnson	230
2024-04-15	David Lee	210
2024-04-20	Emily Chen	250
2024-04-25	Michael Brown	200
2024-05-05	John Smith	190
2024-05-10	Mary Johnson	240
2024-05-15	David Lee	220
2024-05-20	Emily Chen	260
2024-05-25	Michael Brown	210
2024-06-05	John Smith	200
2024-06-10	Mary Johnson	250
2024-06-15	David Lee	230
2024-06-20	Emily Chen	270
2024-06-25	Michael Brown	220
2024-07-05	John Smith	210
2024-07-10	Mary Johnson	260
2024-07-15	David Lee	240
2024-07-20	Emily Chen	280
2024-07-25	Michael Brown	230
2024-08-05	John Smith	220
2024-08-10	Mary Johnson	270
2024-08-15	David Lee	250
2024-08-20	Emily Chen	290
2024-08-25	Michael Brown	240

Solution:

i. **Convert** the Amount Column into **Numeric/General type** by **multiplying with one** Or, using any other **alternative method.**

ii. **Formula-Based Approach:** Use the **UNIQUE** function to get the list of unique customers. Then use the **SUMIF/SUMIFS** function to get the desired result. It is important to **lock the range**.

- =UNIQUE(B2:B41)
- =SUMIFS($D$2:$D$41,$B$2:$B$41,H2)

Customer Name	Total Amount ($)
John Smith	1480
Mary Johnson	1880
David Lee	1720
Emily Chen	2040
Michael Brown	1640

iii. **Non-Formula-Based Approach:** Using **PIVOT TABLE** we can easily aggregate the total amount by each customer.

Row Labels	Total_of_New_Amnt ($)
David Lee	1720
Emily Chen	2040
John Smith	1480
Mary Johnson	1880
Michael Brown	1640
Grand Total	**8760**

3. Power Query

Question: In the given dataset, we can see a column for **Driver's License Number**. The DL is in an **Alphanumeric** state. Where the First two characters refer to the **State Name**, and the last 7 characters represent the **Driver's License Number**. The interviewee needs to **separate the State from the Driver's License number without using any formula or function.**

Person Name	Driver's License Number
John Smith	AB1234567
Mary Johnson	CD2345678
David Lee	EF3456789
Emily Chen	GH4567890
Michael Brown	IJ5678901
Sarah Wilson	KL6789012
Daniel Taylor	MN7890123
Jennifer Davis	OP8901234
Christopher Lee	QR9012345
Amanda Martinez	ST0123456

Solution: Here we need to use **Power Query's Split Non-Digit to Digit** method to get the desired result.

Person Name	State Name	DLN
John Smith	AB	1234567
Mary Johnson	CD	2345678
David Lee	EF	3456789
Emily Chen	GH	4567890
Michael Brown	IJ	5678901
Sarah Wilson	KL	6789012
Daniel Taylor	MN	7890123
Jennifer Davis	OP	8901234
Christopher Lee	QR	9012345
Amanda Martinez	ST	0123456

SQL

1. Difference between DROP, DELETE, and TRUNCATE.

a. DROP:

- **Command Type:** DDL (Data Definition Language)
- **Function:** The DROP command is used to remove an entire table, along with all its data and structure, from the database.
- **Effect:** When we execute a DROP command, the table is permanently deleted from the database, and we cannot retrieve any data from it afterward.
- **Example:** If we have a table named "Customers," executing "DROP TABLE Customers;" will completely remove the Customers table from the database.

b. DELETE:

- **Command Type:** DML (Data Manipulation Language)
- **Function:** The DELETE command is used to remove specific rows or records from a table based on certain conditions.
- **Effect:** When we execute a DELETE command, only the specified rows are removed from the table, leaving the table structure intact.
- **Example:** Executing "DELETE FROM Customers WHERE Age < 18;" will delete all records from the Customers table where the Age is less than 18.

c. TRUNCATE:

- **Command Type:** DDL (Data Definition Language)
- **Function:** The TRUNCATE command is used to remove all the rows from a table while keeping the table structure intact.
- **Effect:** When we execute a TRUNCATE command, all rows are deleted from the table, but the table itself remains, ready to accept new data.
- **Example:** Executing "TRUNCATE TABLE Customers;" will remove all records from the Customers table, but the table structure will remain in place.

Key Differences:

Scope: DROP removes the entire table, DELETE removes specific rows, and TRUNCATE removes all rows but keeps the table.
Data Loss: DROP and TRUNCATE result in the loss of all data, while DELETE allows selective removal.
Efficiency: TRUNCATE is generally faster than DELETE, especially for large tables, as it does not log individual row deletions.
Transaction: DELETE can be rolled back within a transaction, while DROP and TRUNCATE operations cannot be undone.

To conclude, DROP is a DDL command that removes the table entirely, DELETE is a DML command that removes specific rows, and TRUNCATE is a DDL command that removes all rows while preserving the table structure. Each command serves a distinct purpose depending on the requirements of the database operation.

2. Difference between Union and Union All.

The difference between UNION and UNION ALL in SQL lies in how they handle duplicate rows:

a. UNION:

- The UNION operator is used to combine the result sets of two or more SELECT statements into a single result set.
- When we use UNION, duplicate rows are automatically removed from the combined result set. In other words, if a row appears in more than one SELECT statement being UNIONed together, it will only appear once in the final result set.
- The columns in the result set being UNIONed together must have the same data types and be in the same order.

b. UNION ALL:

- The UNION ALL operator also combines the result sets of two or more SELECT statements into a single result set.
- However, unlike UNION, UNION ALL does not remove duplicate rows from the combined result set. It includes all rows from all SELECT statements, even if they are duplicates.
- UNION ALL is generally faster than UNION because it does not need to perform the additional step of removing duplicates.
- Like UNION, the columns in the result sets being UNIONed together with UNION ALL must have the same data types and be in the same order.

In summary:

- **UNION:** Removes duplicate rows from the combined result set.
- **UNION ALL:** Retains all rows from the combined result set, including duplicates.
- Use UNION when you want to eliminate duplicate rows and use UNION ALL when you want to include all rows, including duplicates, and potentially improve performance.

3. Difference between Rank, Dense Rank, and Row Number.

Rank, Dense Rank, and Row Number are window functions in SQL that are used to assign a numerical value to each row based on a specified ordering. However, they differ in how they handle ties (rows with the same value) and the values they assign.

Here's a breakdown of the differences between Rank, Dense Rank, and Row Number:

a. Rank:

- The RANK function assigns a unique rank to each distinct row in the result set based on the specified order.
- If there are ties (rows with the same value), RANK will assign the same rank to each tied row and leave gaps in the ranking sequence. For example, if two rows tie for first place, the next rank will be 3, not 2.
- Example: If three rows have values 10, 10, and 20, the ranks assigned will be 1, 1, and 3 respectively.

b. Dense Rank:

- The DENSE_RANK function also assigns a unique rank to each distinct row in the result set based on the specified order.
- Like RANK, if there are ties, DENSE_RANK will assign the same rank to each tied row. However, it does not leave gaps in the ranking sequence. Instead, it assigns consecutive ranks to tied rows.
- Example: If three rows have values 10, 10, and 20, the ranks assigned will be 1, 1, and 2 respectively.

c. Row Number:

- The ROW_NUMBER function assigns a unique sequential integer to each row in the result set based on the specified order.
- Unlike RANK and DENSE_RANK, ROW_NUMBER does not handle ties. It assigns a distinct number to each row, starting from 1.
- Example: If three rows have values 10, 10, and 20, the row numbers assigned will be 1, 2, and 3 respectively, regardless of ties.

In summary:

Rank: Assigns ranks to rows with gaps in the ranking sequence for ties.
Dense Rank: Assigns ranks to rows without gaps in the ranking sequence for ties.
Row Number: Assigns a unique sequential number to each row without considering ties.

d. What result we will get after the UNION of the below two tables?

id	name	salary
123	Subho	500000
456	Madhu	400000

id	salary
789	600000
159	450000

- We will get an error as to performing union on tables, the tables should have the same structure and the order of the headers/columns must be same. Hence, we will get an error if we perform a UNION on these two tables.

e. What result we will get after doing Inner, Left, Right, Full, and Cross Join on the below two tables?

a	b
1	1
2	1
3	4
NULL	NULL
NULL	NULL
NULL	

Inner Join:

```
select * from A
join B
on a=b;
```

a	b
1	1
1	1

Left Join:

```
select * from A
left join B
on a=b;
```

a	b
1	1
1	1
2	NULL
3	NULL
NULL	NULL
NULL	NULL
NULL	NULL

Right Join:

```
select * from A
right join B
on a=b;
```

a	b
1	1
1	1
NULL	4
NULL	NULL
NULL	NULL

Full Join:

```
select * from A
full join B
on a=b;
```

a	b
1	1
1	1
2	NULL
3	NULL
NULL	NULL
NULL	NULL
NULL	NULL
NULL	4
NULL	NULL
NULL	NULL

Cross Join: (**30 Rows will get returned**, 6x5)

```
select * from A,B;
```

a	b
1	1
2	1
3	1
NULL	1
NULL	1
NULL	1
1	1
2	1
3	1
NULL	1
NULL	1
NULL	1
1	4
2	4
3	4
NULL	4
NULL	4
NULL	4
1	NULL
2	NULL
3	NULL
NULL	NULL
NULL	NULL
NULL	NULL
1	NULL
2	NULL
3	NULL
NULL	NULL
NULL	NULL
NULL	NULL

f. Write a single SQL query to update the gender column, changing 'M' to 'F' and 'F' to 'M'.

- **Query to create the table:**

```
CREATE TABLE emp (
   id INT,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   gender CHAR(1),
   salary DECIMAL(10, 2)
);

INSERT INTO emp (id, first_name, last_name, gender, salary) VALUES
(1, 'John', 'Doe', 'M', 50000.00),
(2, 'Jane', 'Smith', 'F', 60000.00),
(3, 'Michael', 'Johnson', 'M', 55000.00),
(4, 'Emily', 'Brown', 'F', 62000.00),
(5, 'Christopher', 'Jones', 'M', 53000.00),
(6, 'Amanda', 'Davis', 'F', 58000.00),
(7, 'Matthew', 'Miller', 'M', 51000.00),
(8, 'Sarah', 'Wilson', 'F', 59000.00),
(9, 'David', 'Taylor', 'M', 54000.00),
(10, 'Jennifer', 'Anderson', 'F', 61000.00),
(1, 'John', 'Doe', 'M', 50000.00),
(1, 'John', 'Doe', 'M', 50000.00),
(9, 'David', 'Taylor', 'M', 54000.00),
(9, 'David', 'Taylor', 'M', 54000.00);
```

id	first_name	last_name	gender	salary
1	John	Doe	M	50000
2	Jane	Smith	F	60000
3	Michael	Johnson	M	55000
4	Emily	Brown	F	62000
5	Christopher	Jones	M	53000
6	Amanda	Davis	F	58000
7	Matthew	Miller	M	51000
8	Sarah	Wilson	F	59000
9	David	Taylor	M	54000
10	Jennifer	Anderson	F	61000
1	John	Doe	M	50000
1	John	Doe	M	50000
9	David	Taylor	M	54000
9	David	Taylor	M	54000

- **Query to update the gender column:**

```
update emp
set gender= case
                    when gender='M' then 'F'
                    when gender='F' then 'M'
                    end;
```

g.

- **Write a query to find duplicate values in the dataset:**

```
select id,count(id) as cnt from emp
group by id
having count(id)>1;
```

id	cnt
1	3
9	3

- **The process to remove duplicates from the data:**

```
with cte as(
select
*,
ROW_NUMBER() over(partition by id order by id asc) as rnum
from emp)
delete from cte
where rnum>1;
```

- **Process to Add Full_Name Column from the first and last name column:**

```
alter table emp
add full_name varchar(200);

update emp
set full_name=CONCAT(first_name,' ',last_name);
```

Python

1. Load CSV File>Convert the Table into Data Frame Object>Add First and Last Name column and create Full_Name column> Store the File as an Excel file in the system.

- Import Pandas as pd
- Df=pd.read_csv('file_name.csv')
- Df['Full_Name']=df['First_name']+' '+df[last_Name]
- Df.to_excel('new_excel_file.xlsx')

2. Write a function that can return the Reverse Name of the user:

```
def rev_name():
   name=input('Enter your Name: ')
   rev_name=name[::-1]
   return rev_name
```

Power BI

1. Difference between SUM and SUMX:

- **SUM:** It is an aggregate function in DAX (Data Analysis Expressions) used to calculate the sum of values in a column.
- **SUMX:** It is an iterator function in DAX used to perform a calculation for each row in a table or a table expression and then sum up the results. It allows for more complex calculations than SUM, as it iterates over each row, applies a specified expression, and then sums the results.

2. Importance of Data Modeling:

- Data modeling is crucial for organizing and structuring data in a way that facilitates efficient storage, retrieval, and analysis.
- It helps in understanding the relationships between different data entities and improves data integrity and consistency.
- Data modeling ensures that data is stored in a format that meets the requirements of the business and supports analytical processes such as reporting and decision-making.
- It provides a foundation for designing databases, creating relationships between tables, and optimizing queries for performance.

3. Star Schema and Snowflake Schema:

- **Star Schema:** It is a data warehouse schema that consists of one or more fact tables referencing any number of dimension tables. The fact table contains measures or metrics, while dimension tables contain descriptive attributes.

- **Snowflake Schema:** It is a variation of the star schema where the dimension tables are normalized into multiple related tables. This normalization reduces data redundancy but increases complexity. The schema resembles a snowflake with the fact table in the center and dimension tables branching out like snowflakes.

4. Difference between a Fact and a Dimension Table:

Fact Table: A fact table contains quantitative data (measurements or metrics) and is typically the centerpiece of a star schema or snowflake schema. It contains foreign keys that reference dimension tables and numerical values that represent business facts or events (e.g., sales amount, quantity sold).

Dimension Table: A dimension table contains descriptive attributes that provide context for the measures stored in the fact table. Dimension tables are used to filter, group, or categorize the data in the fact table. Examples of dimension tables include customer, product, time, and location tables.

5. Creating a Relationship between Fact Tables:

Fact tables can be related to each other through shared dimensions. If two fact tables share a common dimension (e.g., date dimension), you can create relationships between these tables based on the common dimension. This allows you to perform analysis across multiple fact tables using the shared dimension. Or, we can create a bridge table that contains the Primary Key corresponding to the Foreign Key present in both the Fact Tables.

Created By: Subhajit Dutta **Date Submitted:** 03/04/2024