**DATA ANALYST INTERNSHIP**

**SQL – Diabetes Prediction**

**Q18. Suggest improvements in the database schema to reduce data redundancy and improve data integrity.**

**Ans. To improving a database schema to reduce data redundancy and improve data integrity the following strategies can be used:**

**1. Normalization: This is the process of organizing data in a database to avoid duplication and redundancy. It involves dividing larger tables into smaller ones and defining relationships between them. There are several normal forms, each with a certain level of normalization.**

**2. Use of Primary Keys: Each table should have a primary key column that uniquely identifies each record in the table. This helps ensure the integrity and uniqueness of the data.**

**3. Use of Foreign Keys: These are used to link two tables together. A foreign key in one table points to a primary key in another table, creating a relationship between the two tables. This helps maintain referential integrity.**

**4. Use of Constraints: Constraints such as UNIQUE, NOT NULL, and CHECK can be used to limit the type of data that can go into a table. This ensures that the data adheres to the defined rules, improving data integrity.**

**5. Use of Indexes: Indexes can be used to speed up the retrieval of records on a database table. However, they should be used judiciously as they can slow down data modification operations.**

**6. Proper Use of Data Types: Each column in a table should be assigned the most appropriate data type. This not only saves storage space but also helps ensure that the data stored in the column is of the correct type and format.**

**7. Avoiding Null Values: Whenever possible, avoid allowing null values in your columns. Null values can complicate SQL operations and may lead to unexpected results.**

**Q19. Explain how you can optimize the performance of SQL queries on this dataset.**

**Ans. The performance of SQL queries can be optimized by applying the following:**

**1. Indexing: Consider creating indexes on columns that are frequently queried. For instance, if queries often filter or sort by age, bmi, or diabetes, indexes on these columns could speed up these operations.**

**2. Partitioning: If the table is very large, you might consider partitioning it. For example, you could partition the “diabetes\_prediction” table based on age or diabetes if those columns are often used in queries.**

**3. Optimize Joins: If you’re joining this table with others, ensure you’re using the most efficient type of join and that you’re joining on indexed columns.**

**4. Use Appropriate Data Types: Make sure you’re using the most appropriate data types for each column. For example, if hypertension and diabetes only contain the values 0 and 1, you could use the TINYINT data type to save space.**

**5. Normalization: If there’s redundant data in your table, consider normalizing it. For example, if “EmployeeName” and “Patient\_id” always correspond to the same patient, you could create a separate patients table and reference it in your “diabetes\_prediction” table.**