**Documentation: A Quick Look at MySQL Workbench**

MySQL Workbench is the main tool I use to handle databases and write SQL code. It comes with a full set of features that are necessary for managing and building databases effectively.

**Brief descriptions for each Queries**

1. Creating the Database 'HospitalNDoctor':

**Query**: CREATE DATABASE HospitalNDoctor;

**Description**: This statement creates a new database named 'HospitalNDoctor'. This database is intended to store data related to hospitals and doctors.

1. Selecting the 'HospitalNDoctor' Database for Use:

**Query**: USE HospitalNDoctor;

**Description**: This command selects the newly created 'HospitalNDoctor' database for subsequent operations. All further SQL commands will be executed in this database context.

1. Deleting Existing Tables Named 'Hospital' and 'Doctor':

**Queries**:

DROP TABLE IF EXISTS Hospital;

DROP TABLE IF EXISTS Doctor;

**Description**: These statements check for the existence of tables named 'Hospital' and 'Doctor'. If they exist, these tables are deleted. This ensures that the database does not have old or redundant tables with the same names.

1. Creating the 'Hospital' Table with Specific Columns:

**Description**: This statement creates a new table named 'Hospital' in the 'HospitalNDoctor' database. The table includes three columns: 'Hospital\_Id' (integer type), 'Hospital\_Name' (string type with a maximum length of 255 characters), and 'Bed\_Count' (integer type). These fields are designed to store the hospital's unique identifier, its name, and the count of beds available in the hospital, respectively.

1. Inserting Data into the 'Hospital' Table:

**Query**: Multiple INSERT INTO statements for the 'Hospital' table.

**Description**: These statements are used to insert data into the 'Hospital' table. Each statement adds a new record with three pieces of information: Hospital\_Id (a unique identifier for each hospital), Hospital\_Name (the name of the hospital), and Bed\_Count (the number of beds available in the hospital). The provided data includes:

Record 1: Hospital ID 1, King Edward VII's Hospital, 300 beds.

Record 2: Hospital ID 2, Sheffield Children's Hospital, 600 beds.

Record 3: Hospital ID 3, Birmingham Children's Hospital, 900 beds.

Record 4: Hospital ID 4, Glasgow Royal Infirmary, 1200 beds.

Record 5: Hospital ID 5, St. Pancras Hospital, 1500 beds.

1. Displaying All Data from the 'Hospital' Table:

**Query**: SELECT \* FROM Hospital;

**Description**: This statement retrieves all the data from the 'Hospital' table. The SELECT \* command is used to select all columns, which means it will display every record in the table, showing the Hospital\_Id, Hospital\_Name, and Bed\_Count for each entry.

1. Creating the 'Doctor' Table with Specific Columns:

**Description**: This command creates a new table named 'Doctor' with seven columns. The columns include:

Doctor\_Id: An integer that uniquely identifies each doctor. It's set as the primary key, which means each value in this column must be unique and not null.

Doctor\_Name: A string (up to 255 characters) for the doctor's name.

Hospital\_Code: An integer that likely references a code associated with a hospital.

Joining\_Date: A date field to record when the doctor joined.

Specialty: A string field (up to 255 characters) indicating the doctor's area of specialty.

Salary: A decimal field to represent the doctor's salary, allowing for 10 digits in total, with 2 digits after the decimal point.

Experience: An integer indicating the number of years of experience the doctor has.

1. Inserting Data into the 'Doctor' Table:

**Query**: Multiple INSERT INTO statements for the 'Doctor' table.

**Description**: These statements are used to populate the 'Doctor' table with detailed information about various doctors. Each INSERT INTO statement adds a new record with seven pieces of information: Doctor\_Id (a unique identifier for each doctor), Doctor\_Name (the doctor's name), Hospital\_Code (a reference to the associated hospital), Joining\_Date (the date when the doctor joined), Specialty (the doctor's medical specialty), Salary (the doctor's salary), and Experience (number of years of experience, where null indicates unknown or not applicable). The data includes a diverse range of specialties such as Neurology, Hematology, Cardiology, and others, along with various joining dates and salaries, reflecting a comprehensive database of medical professionals.

1. Displaying All Data from the 'Doctor' Table:

**Query**: SELECT \* FROM Doctor;

**Description**: This command retrieves all the data from the 'Doctor' table. It displays every record, showing each doctor's ID, name, associated hospital code, joining date, specialty, salary, and experience.

1. Joining 'Hospital' and 'Doctor' Tables with Specific Conditions:

**Description**: This statement joins the 'Hospital' and 'Doctor' tables on the condition that the 'Hospital\_Id' from the 'Hospital' table matches the 'Hospital\_Code' in the 'Doctor' table. It then filters the results to show only the data where the hospital ID is 5 and the doctor ID is 002.

1. Selecting Doctors Based on Specialty and Salary Criteria:

**Description**: This query selects all records from the 'Doctor' table where the specialty is 'Endocrinology' and the salary is less than or equal to 4319. It filters the data based on these specific criteria.

1. Displaying Doctors Working in a Specific Hospital:

**Query**: SELECT \* FROM Doctor WHERE Hospital\_Code = '1';

**Description**: This command fetches all records from the 'Doctor' table where the 'Hospital\_Code' is 1. It effectively shows all doctors who are associated with the hospital that has the ID of 1.

1. Updating the Experience of a Specific Doctor:

**Description**: This SQL statement updates the 'Doctor' table by setting the 'experience' field to 10 for the doctor with ID 050. This operation modifies the existing data for that particular doctor's record.