Explanation of SQL Queries:

Create Database:

```
CREATE DATABASE assessment
```

The CREATE DATABASE assessment command creates a new database named "assessment" to store and manage the booking data.

Create Table Bookings:

```
CREATE TABLE bookings (
    Booking_ID VARCHAR(50) PRIMARY KEY,
    Customer_ID INT,
    Customer_Name NVARCHAR(100),
    Booking_Type NVARCHAR(50),
    Booking_Date DATE,
    Status NVARCHAR(50),
    Class_Type NVARCHAR(50),
    Instructor NVARCHAR(100),
    Time_Slot NVARCHAR(50),
    Duration INT,
    Price DECIMAL(10,2),
    Facility NVARCHAR(100),
    Theme NVARCHAR(100),
    Subscription_Type NVARCHAR(50),
    Service_Name NVARCHAR(100),
    Service_Type NVARCHAR(50),
    Customer_Email NVARCHAR(100),
    Customer_Phone NVARCHAR(20)
)
```

The CREATE TABLE bookings statement defines the structure of the "bookings" table.

→Booking_ID is the primary key, ensuring each booking entry is unique.

→Customer-related columns: Includes Customer_ID, Customer_Name, Customer_Email, and Customer_Phone.

→Booking details: Booking_Type, Booking_Date, Status, Class_Type, Instructor, Time_Slot, Duration, and Price store key details about each booking.

→Facility & Service details: Facility, Theme, Subscription_Type, Service_Name, and Service_Type provide additional information about bookings.

Data types:

→NVARCHAR is used for text-based fields to support different character sets.

→DATE is used for Booking_Date.

→DECIMAL(10,2) ensures precise storage for price-related data.

→INT is used for numerical values like ID fields and Duration.

Data Insertion into Database Server:

```
BULK INSERT bookings
FROM 'C:\Users\fayaz\Downloads\DataAnalyst_Assesment_Dataset.csv'
WITH (
    FORMAT = 'CSV',
    FIRSTROW = 2,
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    TABLOCK
)
```

Purpose:

This query imports data from a CSV file into the "bookings" table in SQL Server.

BULK INSERT bookings:

Loads data into the bookings table from a specified file.

FROM 'C:\Users\fayaz\Downloads\DataAnalyst_Assesment_Dataset.csv':

Specifies the file path of the dataset to be imported.

WITH Clause:

FORMAT = 'CSV': Specifies that the file is in CSV format.

FIRSTROW = 2: Skips the first row (assumed to be headers) and starts importing from the second row.

FIELDTERMINATOR = ',': Defines that fields in the CSV file are separated by commas.

ROWTERMINATOR = '\n': Indicates that each row in the CSV is separated by a newline character.

TABLOCK: Improves performance by allowing bulk data insertion with minimal locking.

Handle Missing Values

Fill Missing `Facility` Based on Booking Type

If a **FACILITY** is missing, fill it with the most frequently used facility for that **Booking_Type**.

```
UPDATE boocking b
SET Facility = (
    SELECT TOP 1 Facility
    FROM boocking
    WHERE Booking_Type = b.Booking_Type AND Facility IS NOT NULL
    GROUP BY Facility
    ORDER BY COUNT(*) DESC
)
WHERE b.Facility IS NULL
```

Updating Null Facility Values :

```
UPDATE bookings
SET Facility = Service_Name
WHERE Facility IS NULL
```

→If Facility is missing (NULL), it is replaced with the corresponding Service_Name.

→This ensures that missing facility names are filled with relevant service names.

Standardizing Status Values

```
UPDATE bookings
SET Status = 'Confirmed'
WHERE Status IN ('confirmed', 'CONFIRM', 'Confirm')
```

→These queries standardize the Status values, making them consistent.

```
UPDATE bookings
SET Status = 'Pending'
WHERE Status IN ('PENDING', 'Pending', 'pend')
```

→Converts various forms of "Confirmed" and "Pending" into a uniform format (Confirmed & Pending).

Cleaning Phone Numbers

UPDATE bookings

```
SET Customer_Phone = REPLACE(REPLACE(REPLACE(Customer_Phone, '-', ''), ' ', ''), '(', '')
```

→This query removes unnecessary characters (dashes, spaces, parentheses) from phone numbers.

→Ensures a clean and consistent format for phone numbers.

Updating NULL VALUES in THEME

```
update BOOKINGS
SET Theme  ='No theme'
WHERE theme IS NULL
```

→This query **replaces NULL values** in the **Theme** column with 'No theme'.

→Ensures that every booking has a theme value, avoiding missing data issues in analysis.

Create a Month-Year Column for Filtering in Power BI

To group bookings by month, extract the **Month-Year** format.

ALTER TABLE boocking ADD Booking_Month_Year VARCHAR(20);

UPDATE boocking
SET Booking_Month_Year = FORMAT(Booking_Date, 'MMM yyyy')