7.10

```
CREATE TABLE Hotel (
                        Hotel_No INT PRIMARY KEY,
                        Hotel_Name VARCHAR (100) NOT NULL,
                         City VARCHAR (50) NOT NULL
                        );
```

7.11

```
-- Room table with constraints for type, price, and room number
CREATE TABLE Room (
   Room_No INT CHECK (Room_No BETWEEN 1 AND 100),
   Hotel_No INT,
   Type VARCHAR (50) CHECK (Type IN ('Single', 'Double', 'Family')),
   Price DECIMAL (10, 2) CHECK (Price BETWEEN 50 AND 1000),
   PRIMARY KEY (Room_No, Hotel_No),
   FOREIGN KEY (Hotel_No) REFERENCES Hotel (Hotel_No)
);

-- Guest table with primary key constraint
CREATE TABLE Guest (
   Guest_No INT PRIMARY KEY,
   Guest_Name VARCHAR (100) NOT NULL,
   Guest_Address VARCHAR (150)
);
```

```
-- Booking table with constraints for dates and non-overlapping bookings
CREATE TABLE Booking (
   Hotel_No INT,
   Guest_No INT,
   Date_From DATE CHECK (Date_From > CURRENT_DATE),
   Date_To DATE CHECK (Date_To > Date_From),
   Room_No INT,
   PRIMARY KEY (Hotel_No, Guest_No, Date_From),
   FOREIGN KEY (Hotel_No) REFERENCES Hotel (Hotel_No),
   FOREIGN KEY (Guest_No) REFERENCES Guest (Guest_No),
   FOREIGN KEY (Room_No, Hotel_No) REFERENCES Room (Room_No, Hotel_No),
    -- Constraint to prevent double booking of the same room
   CONSTRAINT unique_room_booking UNIQUE (Hotel_No, Room_No, Date_From),
     -- Constraint to prevent overlapping bookings for the same guest
   CONSTRAINT no_overlap_guest_booking CHECK (
      NOT EXISTS (
         SELECT 1
         FROM Booking b
         WHERE b. Guest_No = Guest_No
          AND b. Hotel_No = Hotel_No
          AND ((Date_From BETWEEN b. Date_From AND b. Date_To)
              OR (Date_To BETWEEN b. Date_From AND b. Date_To))
      )
   )
);
```

7.12

```
CREATE TABLE Booking_Archive AS
SELECT * FROM Booking
WHERE 1 = 0;  -- This creates an empty table with the same structure as Booking

INSERT INTO Booking_Archive
SELECT * FROM Booking
WHERE Date_From < '2023-01-01';

DELETE FROM Booking
WHERE Date_From < '2023-01-01';
```

7.13

```
CREATE VIEW Cheapest_Hotels AS
SELECT H. Hotel_No, H. Hotel_Name, H. City, MIN (R. Price) AS Min_Price
FROM Hotel H
JOIN Room R ON H. Hotel_No = R. Hotel_No
GROUP BY H. Hotel_No, H. Hotel_Name, H. City
ORDER BY Min_Price.
```

7.14

```
CREATE VIEW BRICS_Guests AS
SELECT Guest_No, Guest_Name, Guest_Address
FROM Guest
WHERE Guest_Address LIKE '%Brazil%'
  OR Guest_Address LIKE '%Russia%'
  OR Guest_Address LIKE '%India%'
  OR Guest_Address LIKE '%China%'
  OR Guest_Address LIKE '%South Africa%';
```

Extra-credit

7.20

1. PostgreSQL

- **System Catalog Structure**: PostgreSQL's system catalog is highly structured and organized into schemas under the pg_catalog namespace. Tables within this schema store metadata about tables, columns, indexes, constraints, and more. Key catalog tables include pg_class (tables and indexes), pg_attribute (column information), and pg_index (index metadata).
- **Naming Scheme**: PostgreSQL uses a prefix-based naming convention. For example, catalog tables are prefixed with pg_, such as pg_class, pg_attribute, and pg_index.
- **Object Description Retrieval**: You can query system tables directly for metadata or use built-in views like information_schema.tables. Descriptions can also be retrieved using the COMMENT ON command, which stores and displays comments associated with objects in the pg_description table.

2. MySQL

- **System Catalog Structure**: MySQL's metadata is stored in the information_schema database, which contains tables representing various database objects, such as TABLES, COLUMNS, and STATISTICS for indexes.

- **Naming Scheme**: The naming convention in MySQL's information_schema is intuitive, reflecting object names directly, such as TABLES, COLUMNS, and TRIGGERS.
- **Object Description Retrieval**: To get descriptions, you can query information_schema tables. Additionally, the SHOW command provides descriptions for specific objects (e.g., SHOW TABLE STATUS for table details).

3. Microsoft SQL Server

- **System Catalog Structure**: Microsoft SQL Server stores metadata in the sys schema, which contains tables and views for different database objects. Important tables include sys.tables (for tables), sys.columns (for columns), and sys.indexes (for indexes).
- **Naming Scheme**: SQL Server uses the sys prefix followed by descriptive names, like sys.tables and sys.columns, which makes it easy to identify the purpose of each catalog table.
- **Object Description Retrieval**: SQL Server allows querying sys tables and also provides sp_help and sp_columns stored procedures to retrieve object descriptions in a human-readable format.