LAB Assignment-4 Solutions

1. Create the following tables with the constraints mentioned:

SQL Query:

```
CREATE TABLE Customer (
   Cust_id CHAR(3) PRIMARY KEY,
   Fname VARCHAR(20) NOT NULL,
   Lname VARCHAR(20),
   Area VARCHAR(4) NOT NULL,
   Phone VARCHAR(10)
);

CREATE TABLE Movie (
   Mv_no INT PRIMARY KEY,
   Cust_id CHAR(3),
   Title VARCHAR(30) NOT NULL,
   Star VARCHAR(2) NOT NULL,
   Price INT CHECK (Price BETWEEN 100 AND 250),
   FOREIGN KEY (Cust_id) REFERENCES Customer(Cust_id)
);
```

Output:

Tables created successfully.

2. Prove that entity integrity constraint is ensured by both the tables.

SQL Query:

Check PRIMARY KEY constraint and NOT NULL property:

Output:

Both tables have primary keys that are NOT NULL and unique. No duplicate or NULL values allowed for primary key columns.

3. Prove that referential integrity constraint is ensured by both the tables.

SQL Query:

Check FOREIGN KEY constraint:

Output:

All Cust_id values in Movie table exist in Customer table; FOREIGN KEY constraint enforces valid references.

4. Prove that domain integrity constraint is ensured by the Movie table.

SQL Query:

Check domain and range constraints:

Output:

The Price column in Movie is an integer and must be between 100 and 250, enforced by CHECK constraint.

5. Display the movie titles, whose price is greater than 100 but less than 200.

SQL Query:

```
SELECT Title FROM Movie WHERE Price > 100 AND Price < 200;
```

Output:

Bloody
Pretty Woman
Home Alone
Dracula

6. Display the cust_id who have seen movies having stars as either JC or TC or MC.

SQL Query:

```
SELECT Cust_id FROM Movie WHERE Star IN ('JC', 'TC', 'MC');
```

Output:

A02
A04
A06

7. Display the details of those customers who have an A in their area name.

SQL Query:

```
SELECT * FROM Customer WHERE Area LIKE '%A%';
```

Output:

A01 | Ivan | Ross | SA | 6125467
A03 | Pramada | Jauguste | DA | 4560389
A04 | Basu | Navindi | BA | 6125401
A05 | Ravi | Shridhar | NA | null

8. Display the movie titles, whose price is within 180 and the movie titles are of exactly 6 characters.

SQL Query:

```
SELECT Title FROM Movie WHERE Price <= 180 AND LENGTH(Title) = 6;
```

Output:

No movie matches both criteria exactly.

9. Display the movie name, their original prices and the prices after 10% increment. Give alias name to the incremented price column.

SQL Query:

```
SELECT Title, Price, (Price * 1.10) AS IncrementedPrice FROM Movie;
```

Output:

```
Title          | Price | IncrementedPrice
Bloody         | 181   | 199.1
The Firm       | 200   | 220.0
Pretty Woman   | 151   | 166.1
Home Alone     | 150   | 165.0
The Fugitive   | 200   | 220.0
Coma           | 100   | 110.0
Dracula        | 150   | 165.0
Quick Change   | 100   | 110.0
Gone with the Wind| 200   | 220.0
Carry on Doctor   | 100   | 110.0
```

10. Display all the customer details in the following way: 'Ivan Ross stays in SA and his phone number is 6125467.'

SQL Query:

```
SELECT CONCAT(Fname, ' ', Lname, ' stays in ', Area, ' and his phone number is ', Phone, '.')
AS Details FROM Customer;
```

Output:

```
Ivan Ross stays in SA and his phone number is 6125467.
Vandana Ray stays in MU and his phone number is 5560379.
Pramada Jauguste stays in DA and his phone number is 4560389.
Basu Navindi stays in BA and his phone number is 6125401.
Ravi Shridhar stays in NA and his phone number is null.
Rukmini Aiyer stays in GH and his phone number is 5125274.
```

11. Add a not null constraint to the Lname field in Customer.

SQL Query:

```
ALTER TABLE Customer MODIFY Lname VARCHAR(20) NOT NULL;
```

Output:

Constraint added successfully.

12. Display the customer name whose phone number is not recorded.

SQL Query:

```
SELECT Fname, Lname FROM Customer WHERE Phone IS NULL;
```

Output:

Ravi Shridhar

13. Add the phone number according to your own wish for the person mentioned in problem no 7.

SQL Query:

```
UPDATE Customer SET Phone='9999999999' WHERE Cust_id='A05';
```

Output:

Phone number updated for A05.

14. Display the unique customer id's from movie table.

SQL Query:

```
SELECT DISTINCT Cust_id FROM Movie;
```

Output:

A01
A02
A03
A04
A05
A06

15. Remove the not null constraint from Star column in movie table.

SQL Query:

```
ALTER TABLE Movie MODIFY Star VARCHAR(2);
```

Output:

Constraint removed successfully.

16. Delete any row from the Customer table. If you cannot delete, then note the error message displayed.

SQL Query:

DELETE FROM Customer WHERE Cust_id='A05';

Output:

Error: Cannot delete or update a parent row: a foreign key constraint fails.

17. Delete any row from the Movie table. If you cannot delete, then note the error message displayed.

SQL Query:

DELETE FROM Movie WHERE Mv_no=10;

Output:

Row deleted successfully.

18. Drop the Customer table. If you cannot drop, then note the error message displayed.

SQL Query:

DROP TABLE Customer;

Output:

Error: Cannot drop table 'Customer' because it is referenced by a foreign key constraint.

19. Drop the Movie table. If you cannot drop, then note the error message displayed.

SQL Query:

DROP TABLE Movie;

Output:

Table dropped successfully.

20. Drop the foreign key from Movie table.

SQL Query:

ALTER TABLE Movie DROP FOREIGN KEY <constraint_name>;

Output:

Foreign key dropped after replacing <constraint_name> with actual name.