In SQL, **constraints** are rules or conditions applied to a table's columns to ensure the data in the table meets specific criteria or integrity requirements. Here's an explanation of the constraints mentioned in your example:

1. NOT NULL Constraint:

- The `NOT NULL` constraint ensures that a column cannot contain null values, meaning it must have a value in every row.

- In your example, `ename VARCHAR(30) NOT NULL` specifies that the "ename" column must have a non-null value in each row.

2. AUTO_INCREMENT Constraint (Auto-increment, Identity, or Sequence):

- The `AUTO_INCREMENT` constraint is used to automatically generate unique values for a column, typically used for generating primary key values.

- In your example, `emp_id INT PRIMARY KEY AUTO_INCREMENT` defines the "emp_id" column as the primary key and auto-incremented, which means it will automatically generate unique values for each row.

3. DEFAULT Constraint:

- The `DEFAULT` constraint provides a default value for a column when no value is specified during insertion.

- In your example, `job_desc VARCHAR(20) DEFAULT 'unassigned'` sets the default value for the "job_desc" column as 'unassigned' if no value is provided during insertion.

4. CHECK Constraint:

- The `CHECK` constraint is used to enforce a condition on a column's values. Rows that don't satisfy the condition are rejected.

- In your example, `CHECK (salary > 100000)` ensures that the "salary" column contains values greater than 100,000.

5. UNIQUE Constraint:

- The `UNIQUE` constraint ensures that all values in a column are unique, meaning no duplicate values are allowed.

- In your example, `pan VARCHAR(10) UNIQUE` ensures that the "pan" column contains unique values, typically used for a unique identifier like a PAN (Permanent Account Number).

EXAMPLES

1. Table Creation:

```
CREATE TABLE employee (emp_id INT PRIMARY KEY AUTO_INCREMENT,  ename VARCHAR(30) NOT NULL,  job_desc VARCHAR(20) DEFAULT 'unassigned', salary INT,  pan VARCHAR(10) UNIQUE, CHECK (salary > 100000));
```

- This SQL statement creates a table named "employee" with the following constraints:

- The "emp_id" column is the primary key and auto-incremented.

- The "ename" column is defined as `NOT NULL`, meaning it must have a value in every row.

- The "job_desc" column has a default value of 'unassigned' if no value is provided.

- The "salary" column has a `CHECK` constraint that enforces that the salary must be greater than 100,000.

- The "pan" column has a `UNIQUE` constraint, ensuring that PAN values are unique.

2. Data Insertion:

```
INSERT INTO employee(ename, salary) VALUES ('Prathima', 1000000);
```

```
INSERT INTO employee(ename, salary) VALUES ('Rishad', 10000);
```

- In the first `INSERT INTO` statement, you insert a record for an employee named "Prathima" with a salary of 1,000,000. This insertion is allowed because it meets the "CHECK" constraint (salary > 100,000).

- In the second `INSERT INTO` statement, you attempt to insert a record for an employee named "Rishad" with a salary of 10,000. This insertion is also allowed because it meets the "CHECK" constraint.

3. Data Retrieval:

```
SELECT * FROM employee;
```

- This SQL statement retrieves all rows from the "employee" table. So, it will return both the "Prathima" and "Rishad" records you inserted.