Module 4:
Structured Query Language (SQL)

4.1 Outline:

- Overview of SQL
- Data Definition Commands
- Integrity constraints
  - Key constraints
  - Domain Constraints
  - Referential integrity
  - Check constraints
- Data Manipulation commands
- Data Control commands

Overview of SQL

- SQL stands for **Structured Query Language**
- Used to communicate with a database
- Standard language for relational database management systems
- SQL statements are used to perform different operations on database like retrieval, insertion, updation and deletion of data
- Some common RDBMS that use SQL are MySQL, Oracle, Microsoft Access, Microsoft SQL Server, Sybase, Ingres, etc.
- SQL is developed by IBM as a part of System R project in 1970

Overview of SQL

- Initially it was called as Sequel
- SQL was one of the first commercial languages for Edgar F. Codd's relational model.
- The steps required to execute SQL statements are handled transparently by the SQL database.
- SQL can be characterized as non-procedural

As **procedural languages** the details of the operations to be specified, such as opening and closing tables, loading and searching indexes, and writing data to file systems are required which is not necessary in SQL.

Characteristics of SQL

- SQL is an ANSI and ISO standard computer language for creating and manipulating databases.
- SQL allows the user to create, update, delete, and retrieve data from a database.
- The tokens and syntax of SQL are oriented from English common speech to keep the access barrier as small as possible.

  Hence it is very simple and easy to learn.
- All the keywords of SQL can be expressed in any combination of upper and lower case characters.

It makes no difference whether UPDATE, update, Update, UpDate i.e. the keywords are case insensitive

Characteristics of SQL

- SQL is very powerful language
- SQL works with database programs like DB2, Oracle, MS Access, Sybase, MS SQL Sever etc.

Advantages of SQL

1. High Speed
2. Portable
3. Well Defined Standards Exist
4. Supports object based programming
5. Used with all DBMS systems with any vendor
6. No Coding Required
7. Used for relational databases

Advantages of SQL

8. Easy to learn and understand
9. Complete language for a database
10. Dynamic database language
11. Can be used as programming and interactive language
12. Client/Server language
13. Multiple data views
14. Used in internet

Data types in SQL

- In SQL, we store the data in tabular format where table (relation) is the combination of rows (tuples) and columns (fields).
- While creating table we have to assign data types to the columns.
- These data types are used to decide that which type of data the columns can store.

Data types in SQL

Numeric Data Types

- This data type is used to store a number values that can be decimal or floating point values

a) **Integer number of various sizes**:

- These types of system are used to store natural numbers which are not having any decimal values.
- Example 111, 23 etc.
- As per size of number we can use following types of integers:

i. Interger(p)

ii. Integer or INT

iii. SMALL INT

iv. BIGINT

Data types in SQL

b) **Floating point numbers of various precision**

- This system is used for storing decimal numbers which may be of greater size than integers.
- Example 11.2, 12.3 etc.
- As per size of floating point number we can use following types of numbers.
  i) FLOAT or REAL
  ii) DOUBLE PRECISION

Data types in SQL

c) **Formatted numbers**

- This system used for storing some special numbers which may be of greater size than integers and floating point numbers.

i. DECIMAL or DEC (i, j)

where

- i = Precision = Total number of digits in number.
- j Scale = Total number of digits after decimal point. (default value is 0)
- 12.234 (Decimal(5,3))

ii. NUMERIC (i, j)

- Example 1.12342 (Numeric(1,5))

Data types in SQL

Character string data type

- This data type is used to store a character string which is combination of some alphabets and enclose single quotation marks.
- Example: 'Mahesh', 'abc' etc.

a) **Fixed length:** CHAR (n), Where n= number of characters.

Example abc is stored in char (10) will be stored as 'abc, (abc padded with 7 blank spaces)

b) **Varying length** : VARCHAR (n) Where n = maximum number of characters.

Example It'abc' is stored in VARCHAR (10) will be stored as 'abc ‘(no blank spaces).

Data types in SQL

- **Date time data type**

a) **Date**

- The date data type has 10 positions and its components are YEAR, MONTH and DAY in form in form YYYY-MM-DD
- The length is 10.
- Example Date ‘2009-01-01' (as ‘YYYY-MM-DD’)

b) **Time**

- The TIME data type has at least eight positions, and its components are HOUR, MINUTES and SECONDS in HH:MM:SS
- Example Time 11:16:59' (as HH:MM:SS)

Data types in SQL

c) **Timestamp/ date time**

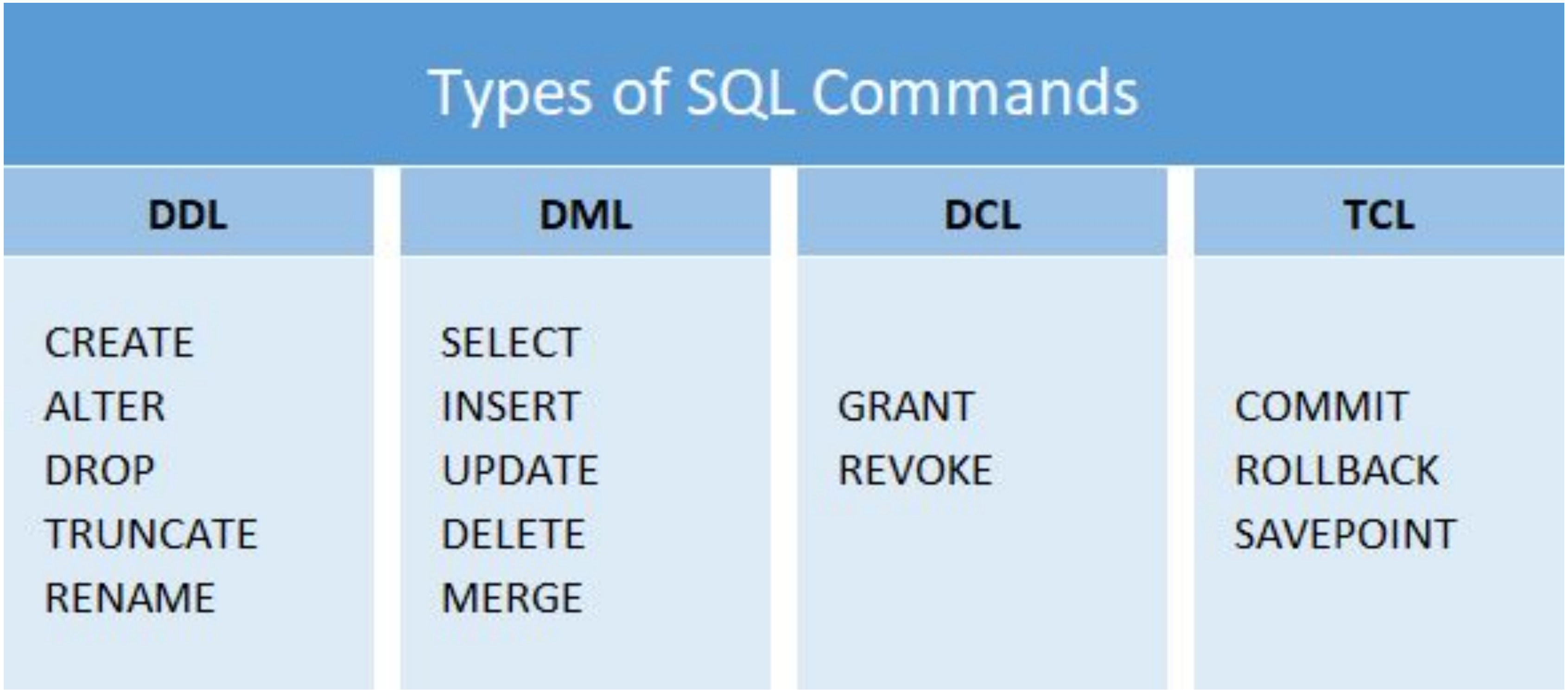
- The TIMESTAMP data type includes both date and time fields
- Represented using the fields YEAR, MONTH, DAY, HOUR, MINUTE and SECOND in the format YYYY-MM DD HH:MM:SS
- Example
  - Timestamp 2009:01:01 11:16:59 648302 (as 'YYYY-MM-DD HH:MM:SS TIMEZONE')
  - CurrentTimeStamp: Local date and time without time zone.

Data types in SQL

d) **Interval**

- This specifies an interval a relative value that can be used to increment or decrement an absolute value of date, time or timestamp.
- INTERVAL YEAR TO MONTH Datatype.
- Example:
- **‘21-5’ Year(2) To Month** indicates an interval of 21 years and 5 months
- **‘21-5’ Year(2)** indicates an interval of 21 years

SQL Languages



Types of SQL Commands

DDL	DML	DCL	TCL
CREATE	SELECT		
ALTER	INSERT	GRANT	COMMIT
DROP	UPDATE	REVOKE	ROLLBACK
TRUNCATE	DELETE		SAVEPOINT
RENAME	MERGE		

Data Definition Language(DDL)

- To create database schema and database objects like table, Data Definition Language is used
- DDL statements are used to build and modify the structure of your tables and other objects in the database
- Set of DDL Commands:
  - **CREATE Statement**: to create database objects
  - **ALTER Statement**: to modify structure of database objects
  - **DROP Statement**: to remove database objects
  - **RENAME Statement**: to rename database objects
  - **TRUNCATE Statement**: to empty database tables
- Database objects are any data structure created in database (Tables, Views)

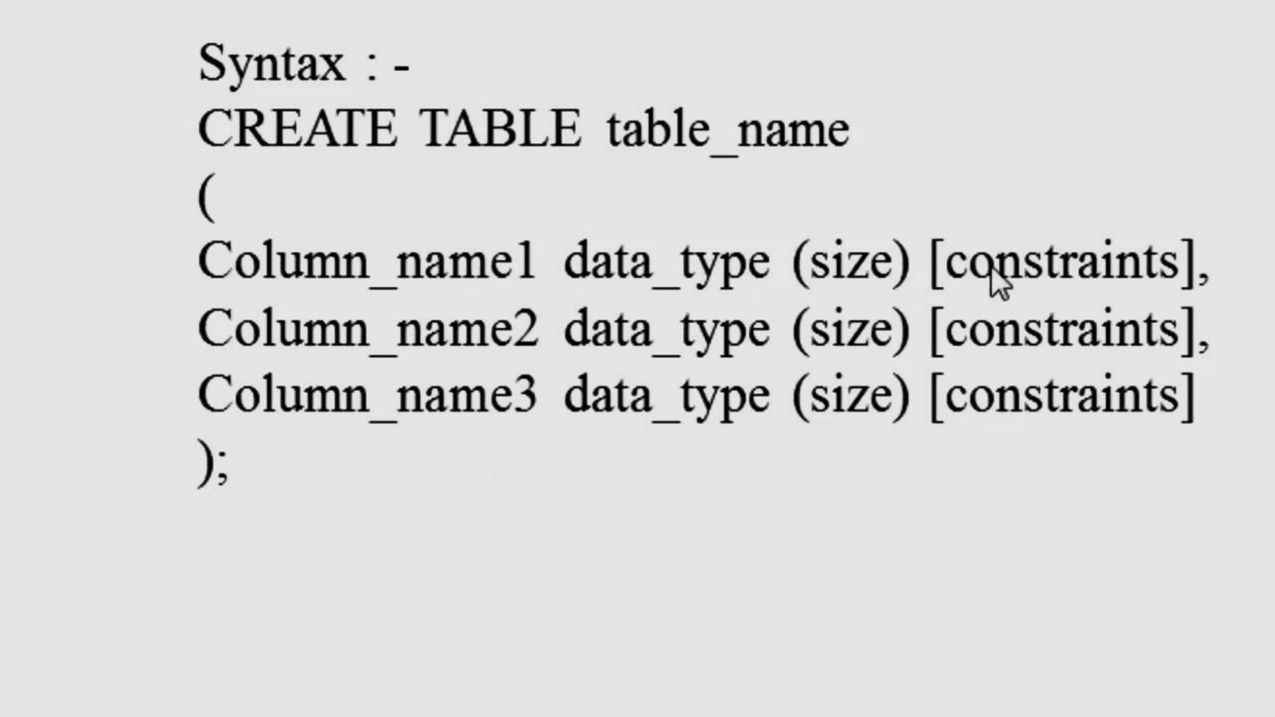
Data Definition Language(DDL)

- When you execute a DDL statements, it takes effect immediately, as it is **auto-committed** into database

Hence no rollback(undo) can be performed with these set of commands

CREATE Command

- CREATE statement is used to create new database objects like table, index and others
- The CREATE TABLE statement is used to create new table with unique name



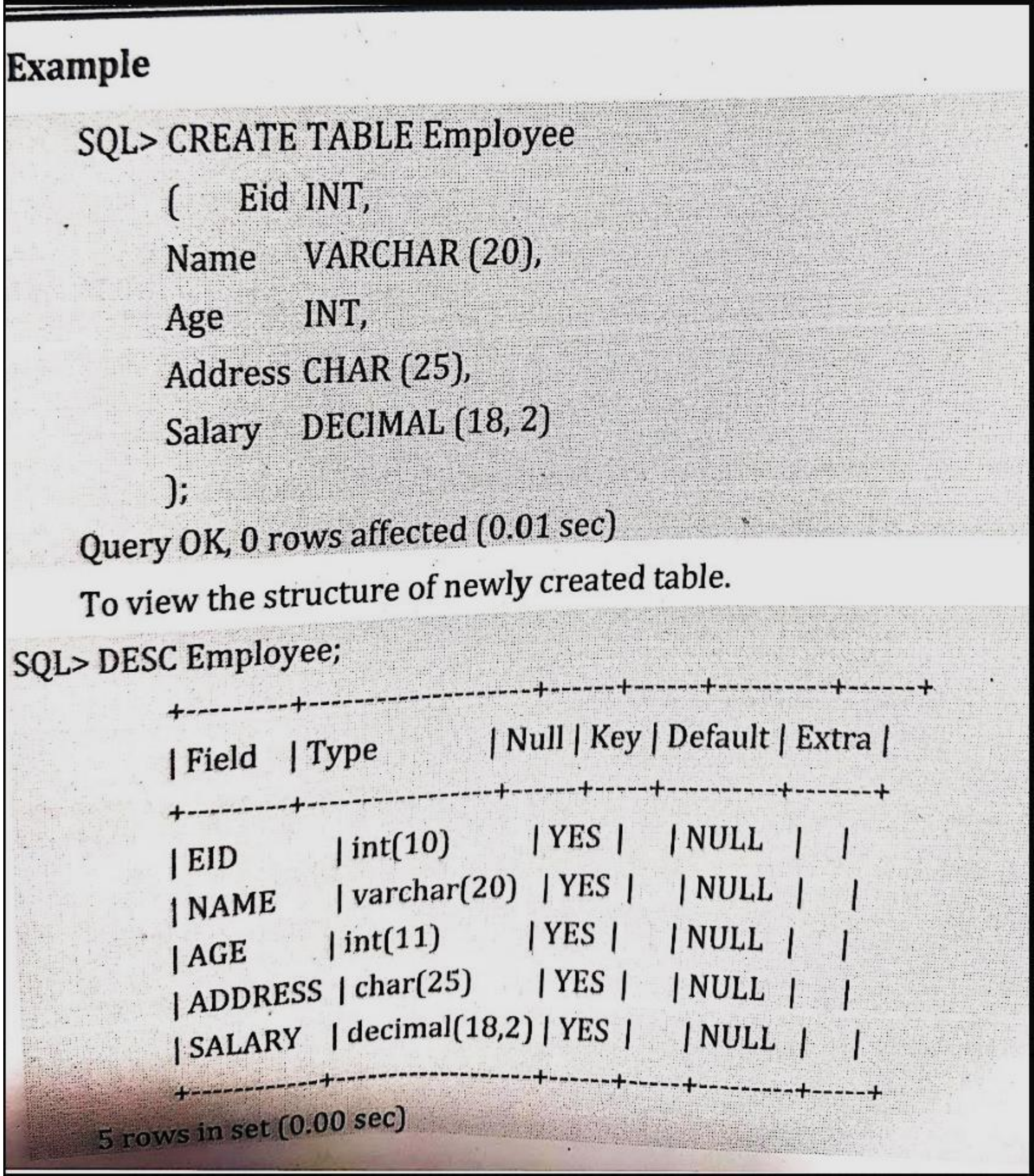
```
Syntax : -
CREATE TABLE table_name
(
Column_name1 data_type (size) [constraints],
Column_name2 data_type (size) [constraints],
Column_name3 data_type (size) [constraints]
);
```

DESC Command

- DESC command is used to display structure of the table
- **Syntax**:

```
DESCRIBE TableName;
        OR
DESC TableName;
```

CREATE Command



Example

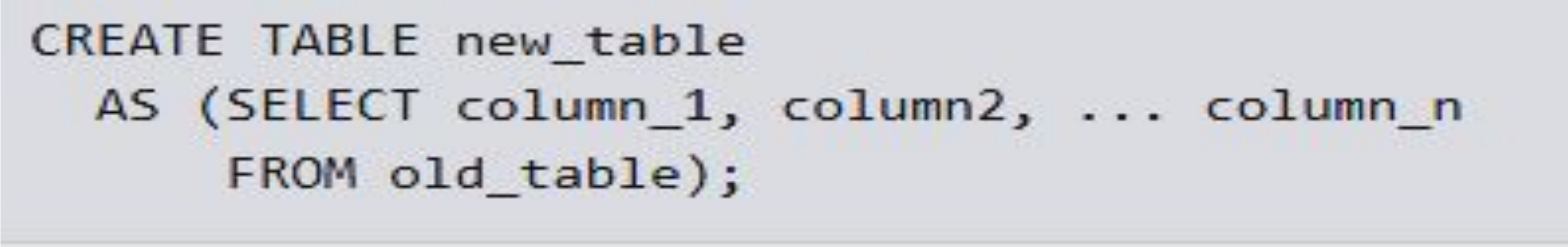
```
SQL> CREATE TABLE Employee
        (    Eid  INT,
        Name     VARCHAR (20),
        Age      INT,
        Address CHAR (25),
        Salary   DECIMAL (18, 2)
        );
Query OK, 0 rows affected (0.01 sec)
```

To view the structure of newly created table.

```
SQL> DESC Employee;
   +---------+-----------------+------+-----+---------+------+
   | Field   | Type            | Null | Key | Default | Extra |
   +---------+-----------------+------+-----+---------+------+
   | EID     | int(10)         | YES |      | NULL    |      |
   | NAME    | varchar(20)     | YES |      | NULL    |      |
   | AGE     | int(11)         | YES |      | NULL    |      |
   | ADDRESS | char(25)        | YES |      | NULL    |      |
   | SALARY  | decimal(18,2)   | YES |      | NULL    |      |
   +---------+-----------------+------+-----+---------+------+
5 rows in set (0.00 sec)
```

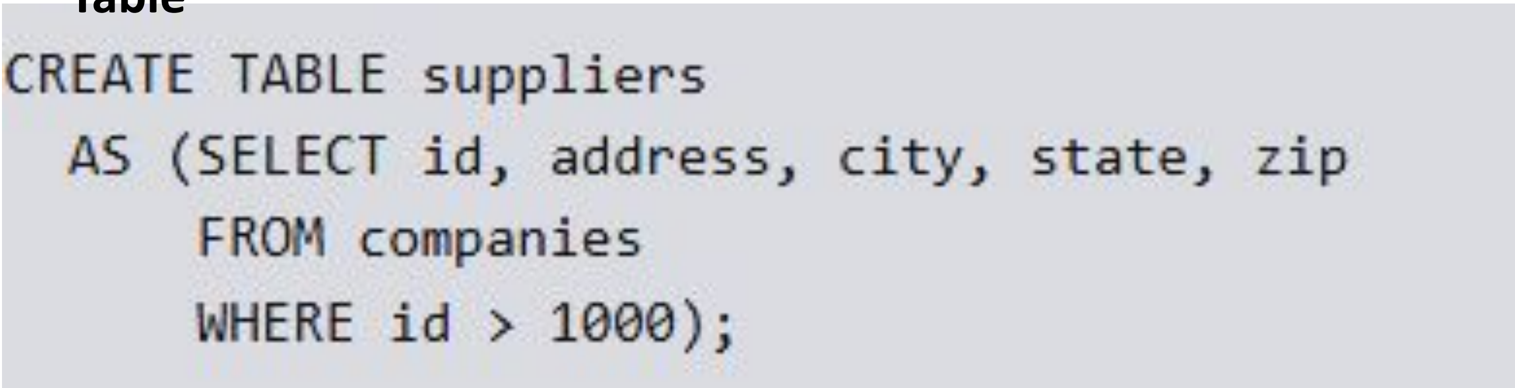
CREATE Command

CREATE Table from Existing Table



```
CREATE TABLE new_table
  AS (SELECT column_1, column2, ... column_n
      FROM old_table);
```

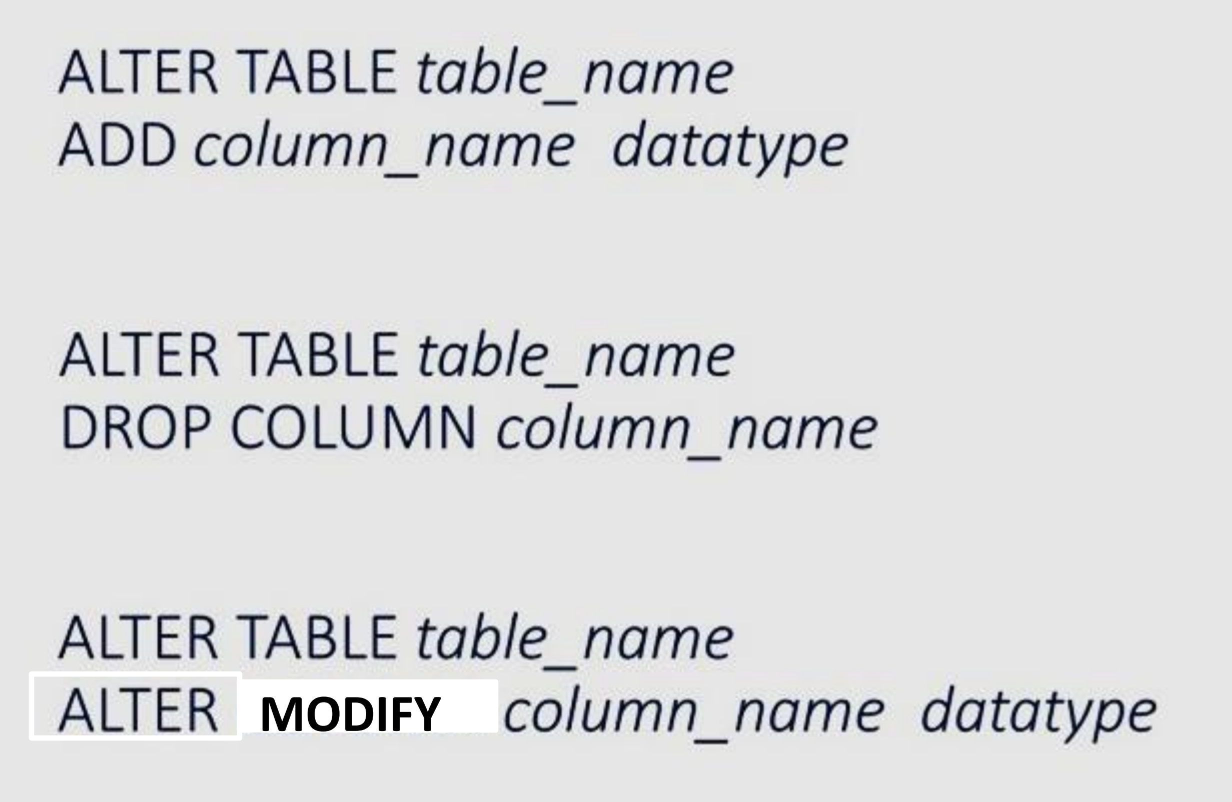
Example: CREATE Table from Existing Table



```
CREATE TABLE suppliers
  AS (SELECT id, address, city, state, zip
      FROM companies
      WHERE id > 1000);
```

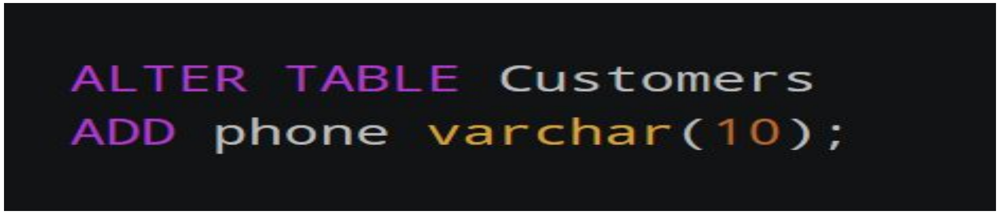
ALTER Command

- Once database object is created in database, we may require ALTER command to update structure of database object
- The ALTER Statement can be used to add, delete or modify columns in existing table

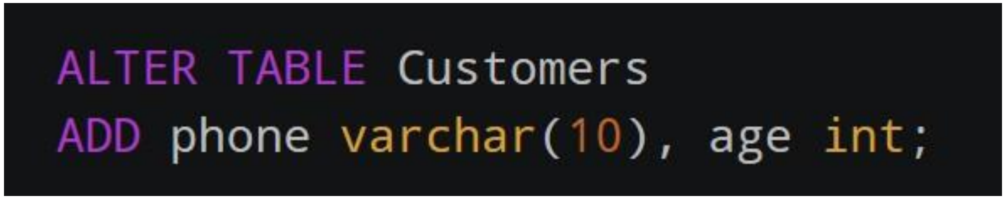


```
ALTER TABLE table_name
ADD column_name  datatype


ALTER TABLE table_name
DROP COLUMN column_name


ALTER TABLE table_name
ALTER  MODIFY  column_name  datatype
```

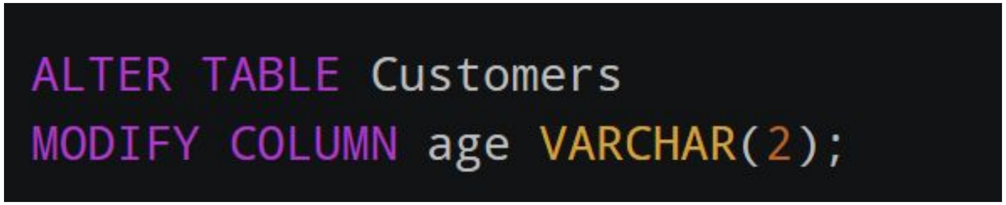
ALTER Command



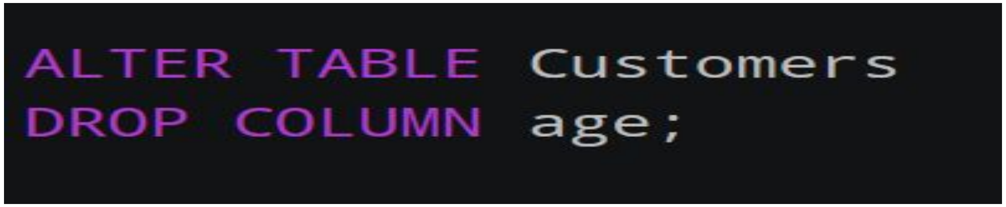
```
ALTER TABLE Customers
ADD phone varchar(10);
```



```
ALTER TABLE Customers
ADD phone varchar(10), age int;
```



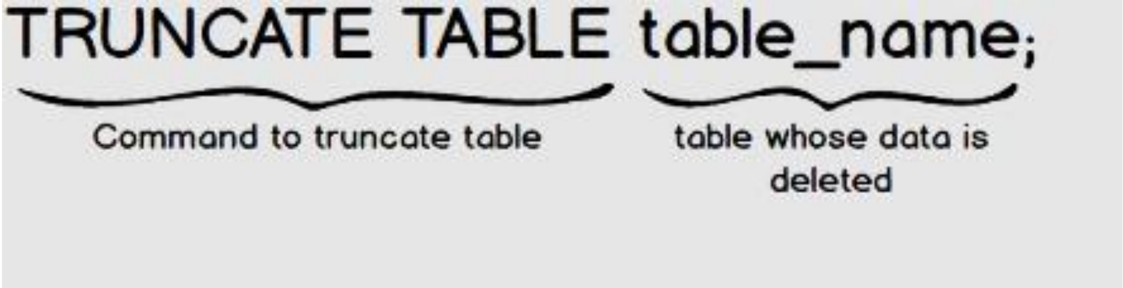
```
ALTER TABLE Customers
MODIFY COLUMN age VARCHAR(2);
```



```
ALTER TABLE Customers
DROP COLUMN age;
```

TRUNCATE Command

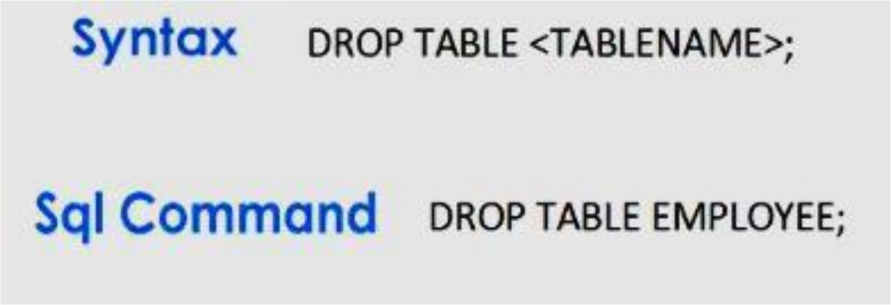
- This command is used to **delete all records from existing table**
- **It is possible to do same action with DROP TABLE command but it would remove complete table structure from the database**
- A DELETE command will also remove all data from table but with DELETE data deletion can be rolled back and truncate acts as permanent data deletion with no roll back possible
- Truncate will de-allocates memory space. So that free space can be used by other tables unlike DELETE command



```
TRUNCATE TABLE table_name;
```

DROP Command

- DROP command can be used to remove database objects from user database
- DROP TABLE statement is used to remove table definition and all related data like indexes, triggers, constraints and permissions for the table
- Developer must be careful while running this command because once table is dropped then all information available in that table will also be lost forever and no rollback can be done



Syntax

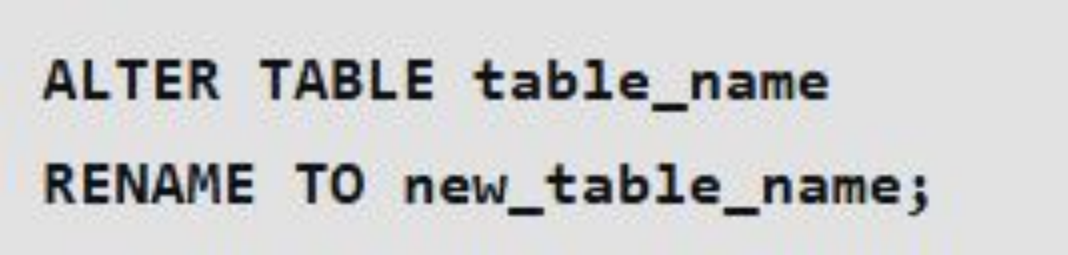
```
DROP TABLE <TABLENAME>;
```

Sql Command

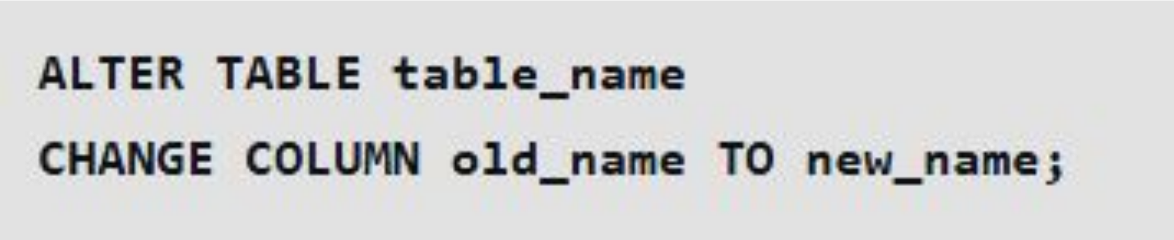
```
DROP TABLE EMPLOYEE;
```

Rename Command

- It is possible to change name of table with or without data in it using simple RENAME command
- We can rename table object at any point in time



```
ALTER TABLE table_name
RENAME TO new_table_name;
```



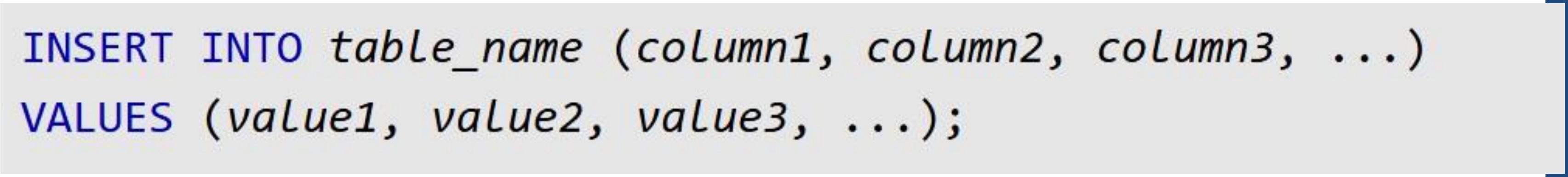
```
ALTER TABLE table_name
CHANGE COLUMN old_name TO new_name;
```

Data Manipulation Language(DML)

- Data Manipulation Language statements are used for manipulating data in database
- DML commands are not auto-committed like DDL Statements Changes done by DML commands can be rolled back
- Under DML Commands we perform:
  - **INSERT Statement**
  - **UPDATE Statement**
  - **DELETE Statement**
  - **SELECT Statement**

Insert Statement

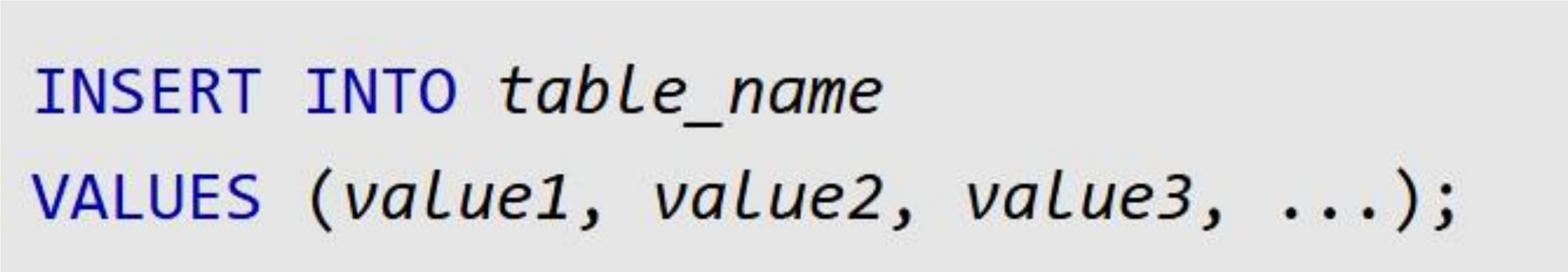
- INSERT Statement used to add records to existing table
- To insert data into table, SQL INSERT INTO command can be used
- To insert few values in table as per columns names we can use following
- **Syntax:**



```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Insert Statement

- If all values for all the columns of the table are to be added then **no need to specify** the column names in SQL Query
- However, make sure the order of the values is in the same order as the columns in the table.
- Here, **syntax** would be as follows:



```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Insert Statement

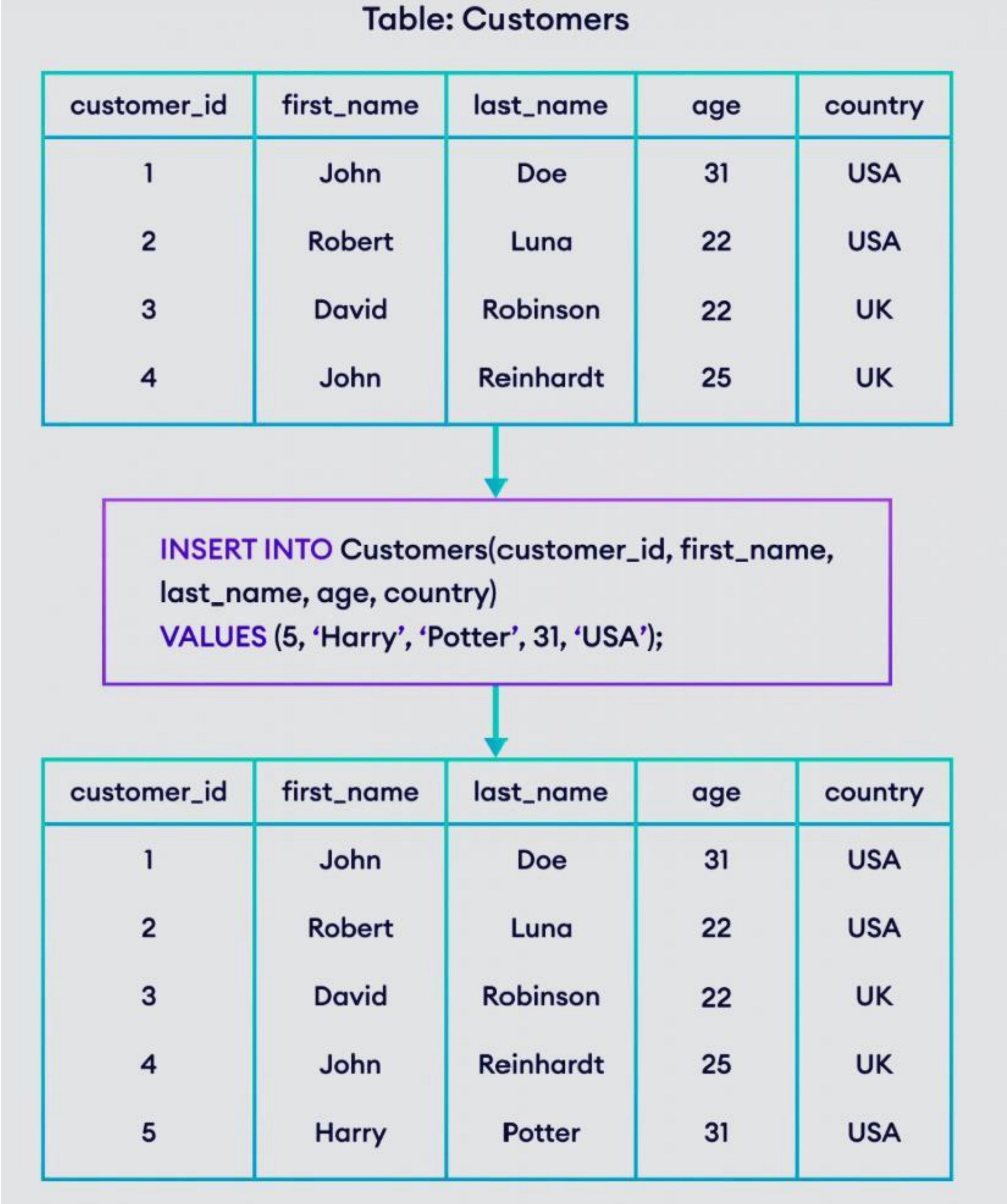


Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK

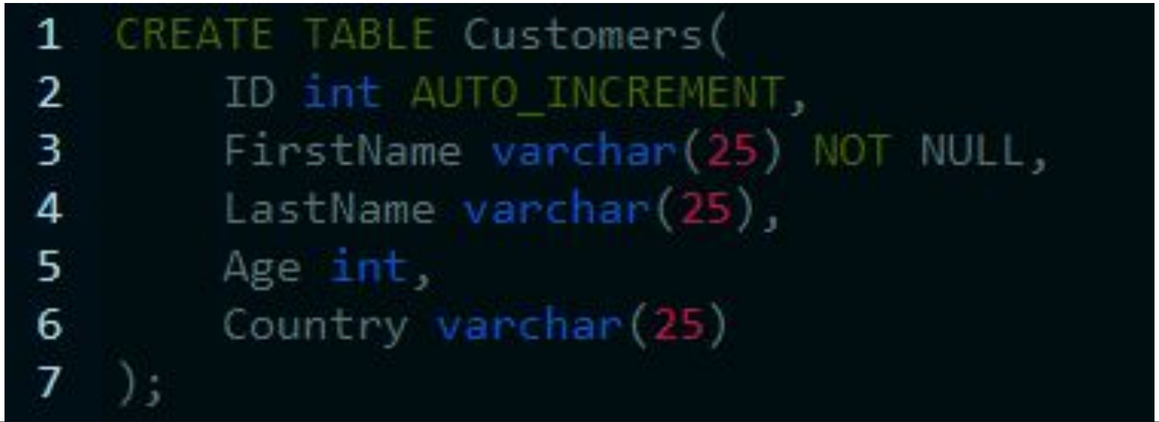
```
INSERT INTO Customers(customer_id, first_name,
last_name, age, country)
VALUES (5, ‘Harry’, ‘Potter’, 31, ‘USA’);
```

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Harry	Potter	31	USA

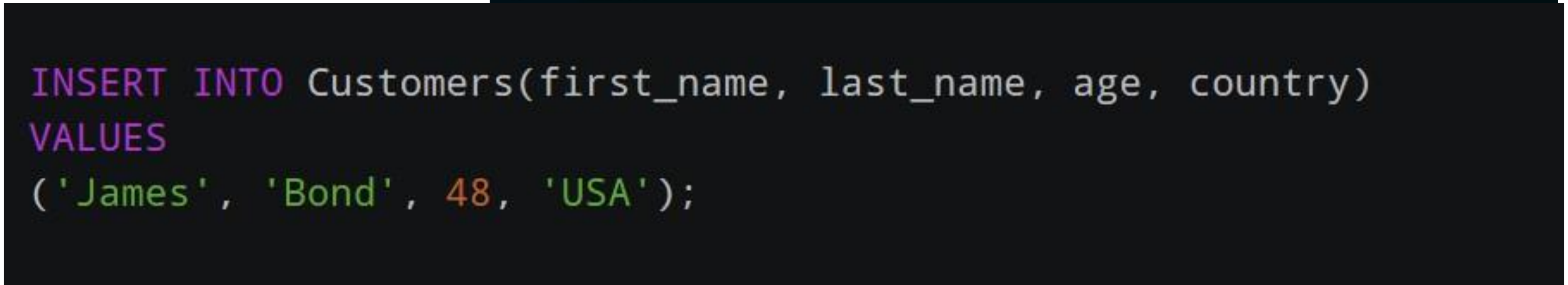
Insert Statement

Insert Row Providing Value Explicitly

- It's possible to provide default values to a column (for example, auto incrementing a column). In a database table, the **ID** field is usually unique auto incremented.
- In such cases, we can omit the value for that column during row insertion
- For example,



```
CREATE TABLE Customers(
    ID int AUTO_INCREMENT,
    FirstName varchar(25) NOT NULL,
    LastName varchar(25),
    Age int,
    Country varchar(25)
);
```

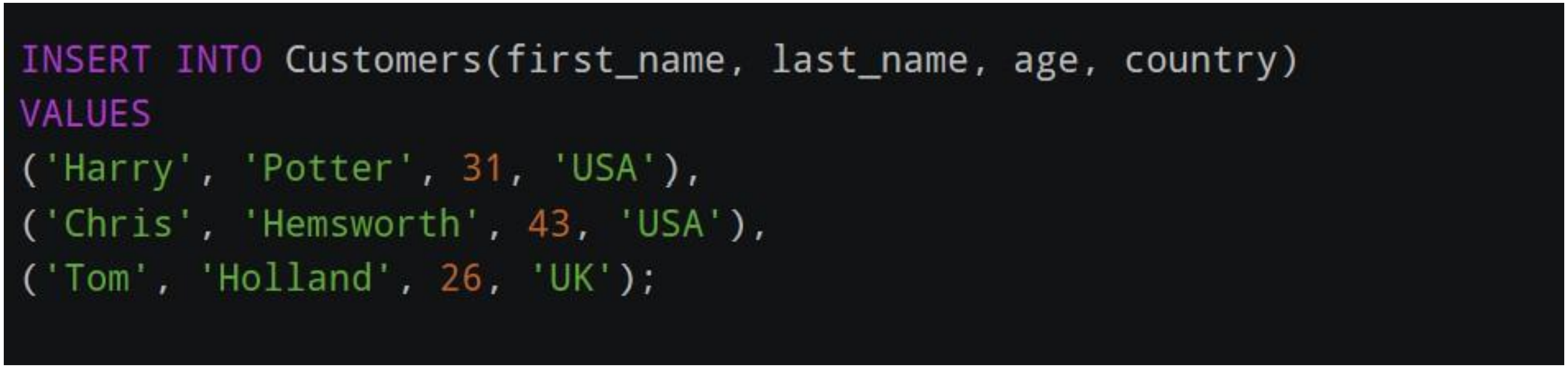


```
INSERT INTO Customers(first_name, last_name, age, country)
VALUES
('James', 'Bond', 48, 'USA');
```

Insert Statement

Insert Multiple Rows at Once in SQL

- It's also possible to insert multiple rows to a database table at once.
- For example,

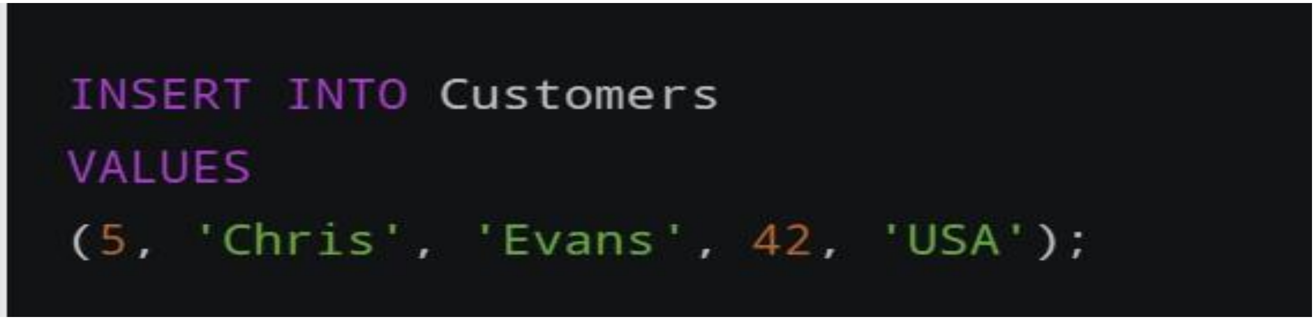


```
INSERT INTO Customers(first_name, last_name, age, country)
VALUES
('Harry', 'Potter', 31, 'USA'),
('Chris', 'Hemsworth', 43, 'USA'),
('Tom', 'Holland', 26, 'UK');
```

Insert Statement

Insert rows Without Specifying Column Names

- It is also possible to insert values in a row without specifying column names.
- For example

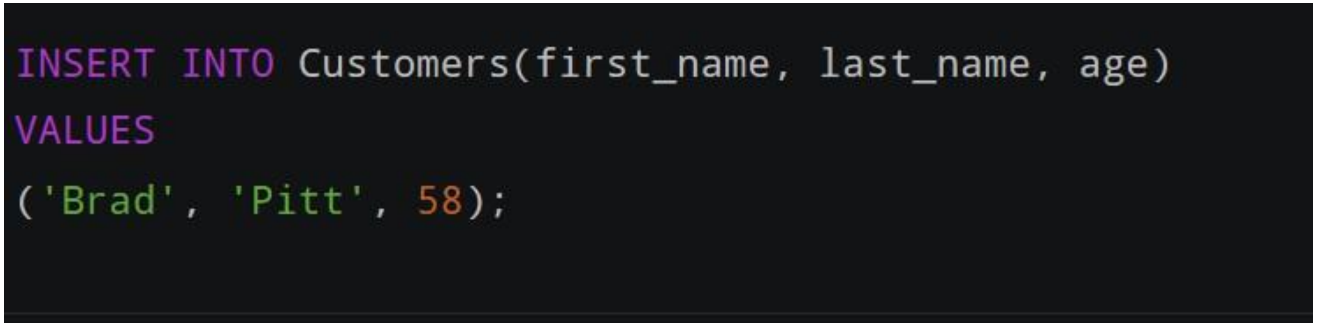


```
INSERT INTO Customers
VALUES
(5, 'Chris', 'Evans', 42, 'USA');
```

Insert Statement

Not Including All Columns During Insertion

- If we skip column names during row insertion, the values of those columns will be NULL



```
INSERT INTO Customers(first_name, last_name, age)
VALUES
('Brad', 'Pitt', 58);
```

Update Statement

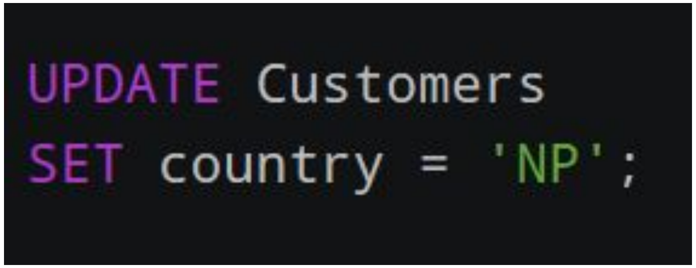
- Update statement is used to modify the existing data present in the table
- To update data in table, SQL UPDATE command can be used
- To update all rows in table we can use following
- <u>**Syntax:**</u>

UPDATE <Table_name>

SET

column1=new_value

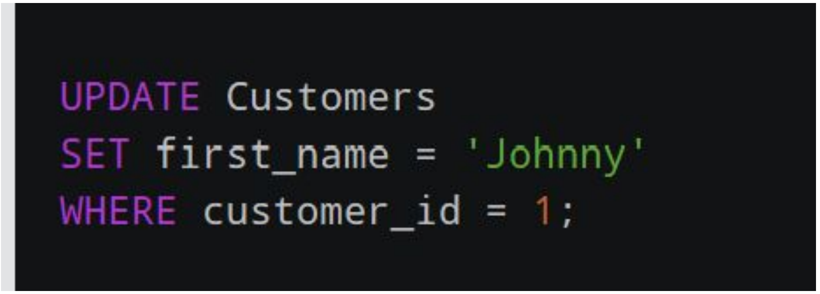
- For example,



```
UPDATE Customers
SET country = 'NP';
```

Update Statement

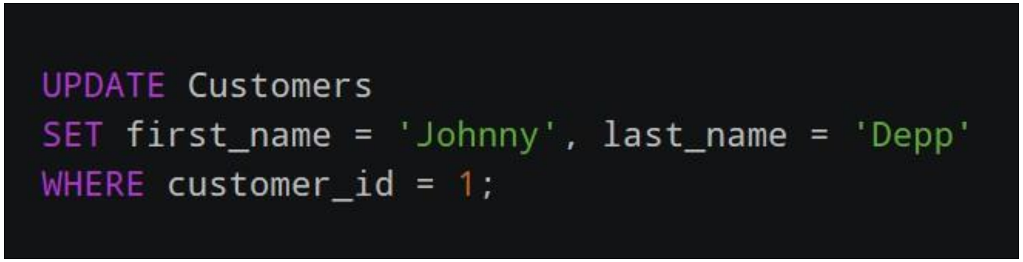
- The SQL UPDATE statement is used to edit existing rows in a database table. For example,



```
UPDATE Customers
SET first_name = 'Johnny'
WHERE customer_id = 1;
```

Update Multiple Values in a Row

- We can also update multiple values in a row at once.
  For example,

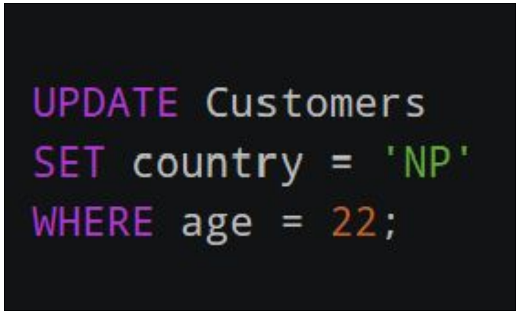


```
UPDATE Customers
SET first_name = 'Johnny', last_name = 'Depp'
WHERE customer_id = 1;
```

Update Statement

Update Multiple Rows

- The UPDATE statement can update multiple rows at once.
- For example,



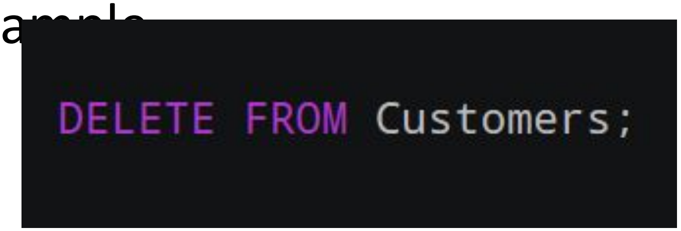
```
UPDATE Customers
SET country = 'NP'
WHERE age = 22;
```

Delete Statement

- DELETE Statement is used to delete some or all records from existing table
- To delete data into a table, SQL DELETE command can be used

Delete all Rows in a Table

- The WHERE clause determines which rows to delete. However,
  we can delete all rows at once if we omit the WHERE clause.
- Syntax: DELETE FROM <TABLE_NAME>
- For example



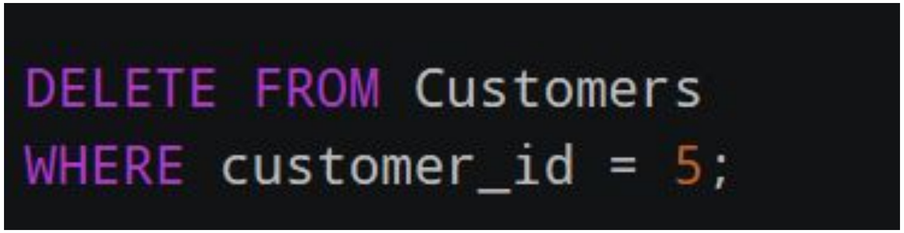
```
DELETE FROM Customers;
```

Delete Statement

- In SQL, we use the DELETE statement to delete specific row(s) from a database table.
- **Syntax**:

DELETE FROM <TABLE_NAME> WHERE<Condition>

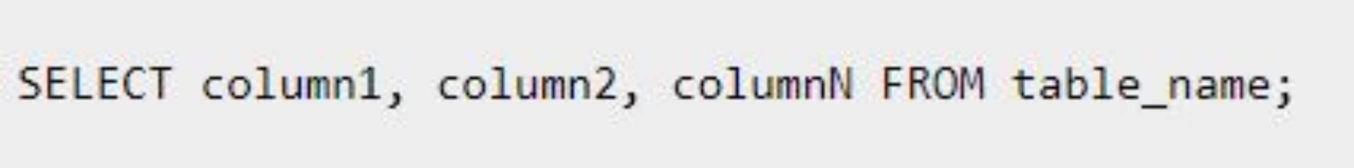
- For example,



```
DELETE FROM Customers
WHERE customer_id = 5;
```

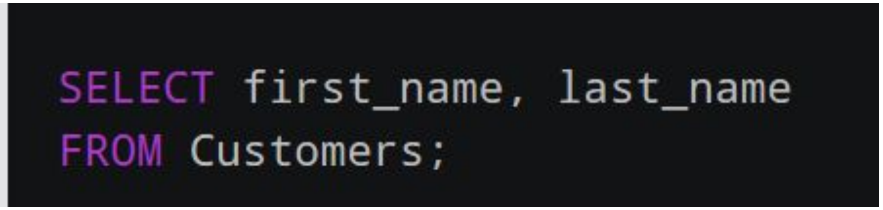
SELECT Statement

- Select statement is used to retrieve data from database.
- SELECT query can never make any change in the database.
- The data returned by the SELECT query is in the form of result sets
- **Syntax**:



```
SELECT column1, column2, columnN FROM table_name;
```

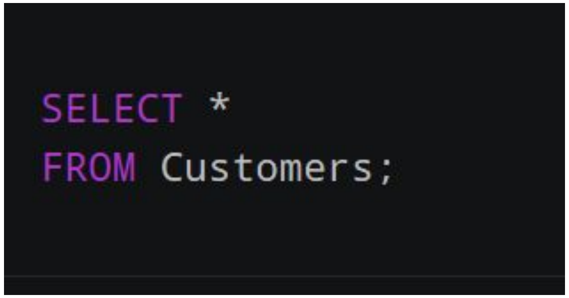
- The SQL SELECT statement is used to select (retrieve) data from a database table. For example,



```
SELECT first_name, last_name
FROM Customers;
```

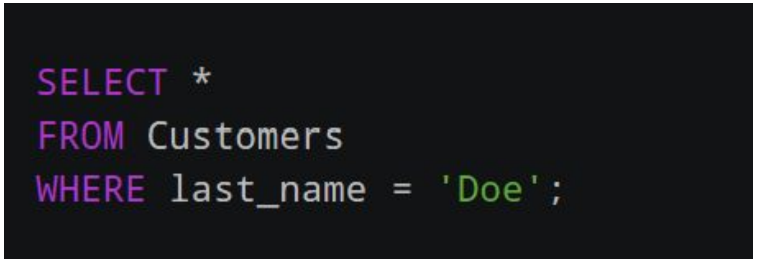
SELECT Statement

- To select all columns from a database table, we use the * character. For example,



```sql
SELECT *
FROM Customers;
```

- A SELECT statement can have an optional WHERE clause. The WHERE clause allows us to fetch records from a database table that matches specified condition(s). For example,



```sql
SELECT *
FROM Customers
WHERE last_name = 'Doe';
```

Data Control Language(DCL)

- DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.
- DCL is set of commands used to
  - **GRANT**- Gives user's privileges to perform task on database.
  - **REVOKE**-Withdraw user's privileges given by using the GRANT command.

Data Control Language(DCL)

Privileges:

- The set of actions that a user can perform on a database object are called the **privileges**
- Privilege is right to execute particular SQL statement on the database
- The high level user(DBA) has power to grant access to database and its objects
- Privileges can be of many types:
    - **System Privileges**: creating table
    - **Object Privileges**: execute query on table object
    - **Ownership Privileges**: execute query on tables created by same user

Data Control Language(DCL)

1. **System Privileges:**

- Rights and restriction implemented on database to control which user can access how much data in the database
- User requires system privileges to gain access to database
- System privileges are generally provided by DBA
- Few system privileges are as follows

System Privileges	Authorized to
CREATE USER	Create number of users in DBMS
DROP USER	Drop any other users in DBMS
CREATE ANY TABLE	Create table object in any schema
SELECT ANY TABLE	Query table object or view in any schema
DROP ANY TABLE	Drop table object in any schema

Data Control Language(DCL)

2. **Object privileges**

- Rights and restrictions to change contents of database objects.
- User requires object privileges to manipulate the content of object within database.
- Not all database users are allowed to make such changes in database; hence administrator should have control over all objects modification.
- The user which has GRANT ANY PRIVILEGE system privilege can act like administrator to control database modifications.
- Different objects has different privileges assigned for him.

Data Control Language(DCL)

- Few object privileges are as follows:

Object Privileges	Authorized To
SELECT	Select rows from table or view
INSERT	Add new rows to table or view
DELETE	Remove some rows from table or view
UPDATE	Modify content of rows from table or view
EXECUTE	To run procedure
REFERENCES	To reference a particular table using foreign key and check constraint

Data Control Language(DCL)

3. **Ownership privileges**

- Whenever you create a database object (like table or view) with the CREATE statement, you will become its owner and you will get full privileges for the table. (Like SELECT, INSERT, DELETE, UPDATE, and all other privileges).
- All other users are having no privileges on the newly created database object.
- You as owner of database object can explicitly give grant privileges to any other user by using the GRANT statement

Data Control Language(DCL)

Granting Privileges

- A system privilege is the right to perform a particular action, or to perform an action on any schema objects of a particular type.
- An authorized user may pass on this authorization to other users.

  This process is called as **granting of privileges**
- Generally GRANT statement is used by owner of table or view to give other users access permissions
- In SQL user accounts must be present in system before we can grant privileges to him.

Data Control Language(DCL)

Syntax:

GRANT <ALL | privilege list>
ON <Table_name or view_name>
TO <user| role list| PUBLIC>
[WITH GRANT OPTION]

Privilege List	Meaning
ALTER	Tables and Views
CREATE	Tables and Views
DROP	Tables and Views
DELETE	Tables and Views
INSERT	Tables and Views
SELECT	Tables and Views
UPDATE	Tables and Views
ALL	Tables and Views

[WITH GRANT OPTION]
It is used to allow user to grant privileges (which are granted to him) to other users.

Data Control Language(DCL)

Example

- Consider an example for granting update authorization to the Emp_Salary relation of the company database
- Assume that initially DBA grants update authorization on Emp_Salary to other users U1, U2, U3
- The following grant statement grants user U1, U2 and U3 the select privilege on Emp_Salary relation

```
GRANT SELECT, INSERT
ON mydb.*
TO mahesh'@'somehost;
```

Data Control Language(DCL)

- Following grant statement gives users all authorization on Emp_Salary relation using public keyword;

a) **Database privileges**

```
GRANT ALL
ON *.*
TO 'mahesh'@'somehost;
```

b) **Column privileges**

- This privilege authorizes a user to execute a function or procedure.

```
GRANT SELECT (col1), INSERT (col1, col2)
ON mydb.mytbl
TO 'mahesh'@'somehost'
```

Data Control Language(DCL)

c) **Table privileges**

This privilege authorizes a user to execute a function or procedure.

```
GRANT ALL
ON mydb.mytbl
TO 'mahesh'@'somehost';
```

OR

```
GRANT SELECT, INSERT
ON mydb.mytbl
TO 'mahesh'@'somehost';
```

Data Control Language(DCL)

Revoking Privileges

- We can reject privileges given to particular user with the help of revoke statement
- To revoke an authorization we use the revoke statement
- **Syntax:**

REVOKE <ALL | privileges list>
ON <relation name or view name>
FROM <user| role list | PUBLIC>
[RESTRICT/ CASCADE]

CASCADE: This will revoke all privileges along with all dependent grant privileges.

RESTRICT: This will not revoke all related grants only removes that GRANT only

Data Control Language(DCL)

Example:

- The revocation of privileges from user or role may cause other user or roles also have to leave that privilege. This behavior is called cascading of the revoke.

a) **To remove select privilege from users U1, U2 and U3**.

```
REVOKE SELECT
ON mydb.mytbl
FROM 'mahesh'@'somehost;
```

b) **To remove update rights on amount column of Emp_Salary from U1, U2 and U3.**

```
REVOKE UPDATE (amount)
ON EmpSalary
FROM 'mahesh'@'somehost
```

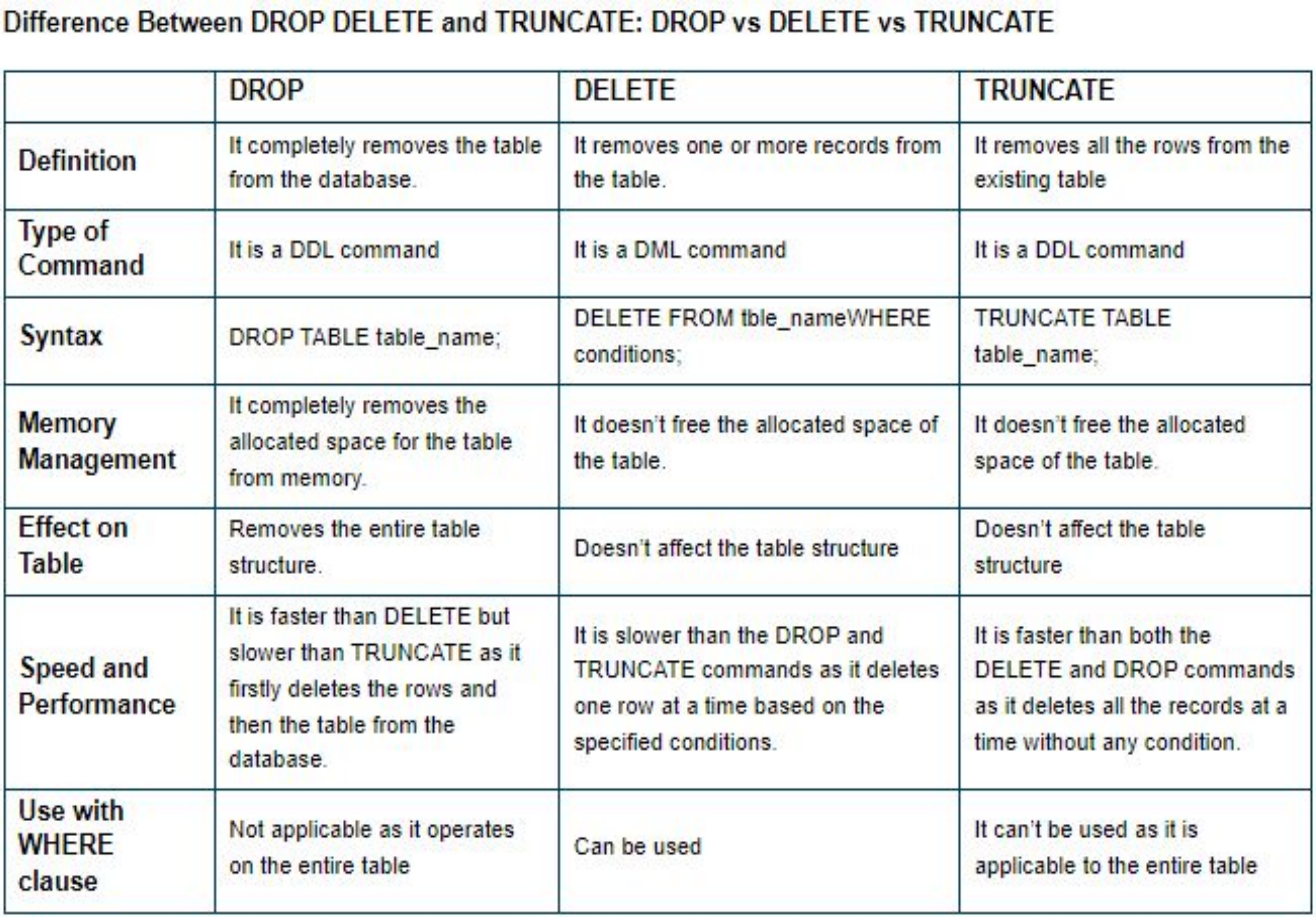
Data Control Language(DCL)

c) **To remove reference right on amount column from user U1.**

```
REVOKE REFERENCES (amount)
ON Emp_Salary
FROM 'mahesh'@'somehost;
```

The revoke statements may alternatively specify restrict if we don't want cascade behavior.

```
REVOKE SELECT
ON Emp_Salary
FROM 'mahesh'@'somehost
RESTRICT;
```



Difference Between DROP DELETE and TRUNCATE: DROP vs DELETE vs TRUNCATE

	DROP	DELETE	TRUNCATE
Definition	It completely removes the table from the database.	It removes one or more records from the table.	It removes all the rows from the existing table
Type of Command	It is a DDL command	It is a DML command	It is a DDL command
Syntax	DROP TABLE table_name;	DELETE FROM tble_nameWHERE conditions;	TRUNCATE TABLE table_name;
Memory Management	It completely removes the allocated space for the table from memory.	It doesn't free the allocated space of the table.	It doesn't free the allocated space of the table.
Effect on Table	Removes the entire table structure.	Doesn't affect the table structure	Doesn't affect the table structure
Speed and Performance	It is faster than DELETE but slower than TRUNCATE as it firstly deletes the rows and then the table from the database.	It is slower than the DROP and TRUNCATE commands as it deletes one row at a time based on the specified conditions.	It is faster than both the DELETE and DROP commands as it deletes all the records at a time without any condition.
Use with WHERE clause	Not applicable as it operates on the entire table	Can be used	It can't be used as it is applicable to the entire table

Transaction Control Language(TCL)

- Any SQL query can be executed with two basic operations on the database objects:
  - Read
  - Write
- After executing SQL query we must specify its final action as **commit** (save data) or **abort** (or revert back changes).
- The COMMIT statement ends the operations and makes all changes made to the data permanent on successful completion
- ABORT terminates and undoes all the actions done so far.

Transaction Control Language(TCL)

- TCL(Transaction Control Language) commands deals with the transaction within the database.
- Examples of TCL Commands:
  - **COMMIT**- Commits a Transaction
  - **ROLLBACK**- Rollbacks a transaction in case of any error occurs
  - **SAVEPOINT**- Sets a savepoint within a transaction

SAVEPOINT and ROLLBACK Command:

- SAVEPOINT is an indicator inside a transaction that is used for a partial rollback.
- When we are doing change to a transaction, we can create SAVEPOINTs to mark different points within the transaction.

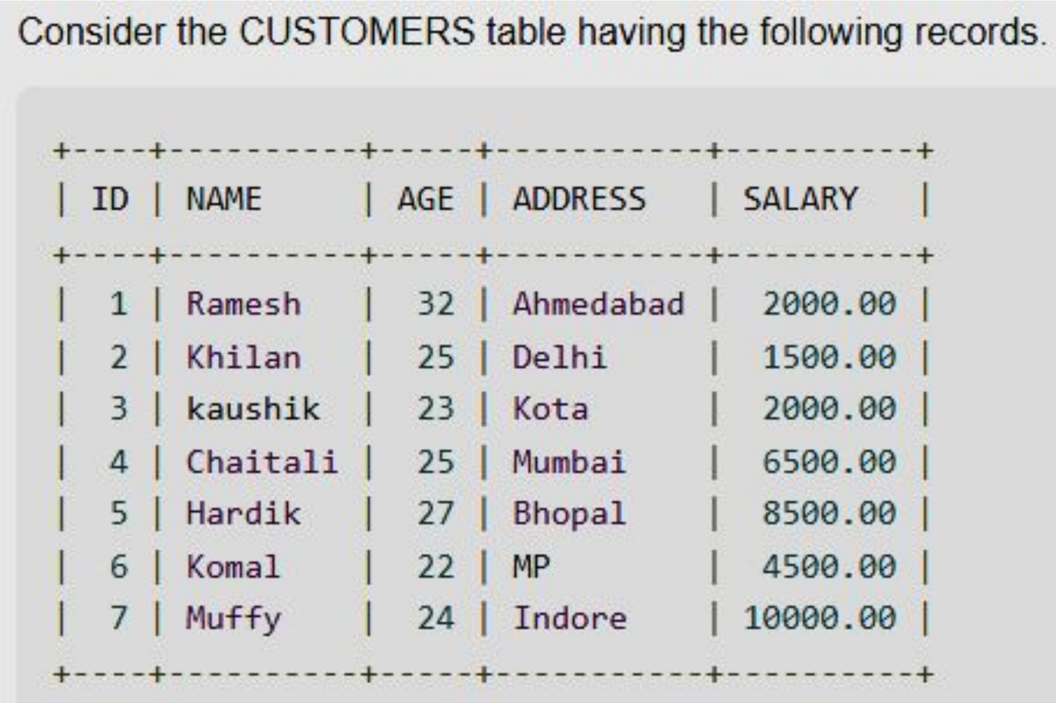
Transaction Control Language(TCL)

- If at some stage we realize that an error is generated, then we can rollback up to a SAVEPOINT which we already created inside transaction.
- **Name of savepoint should be unique inside the transaction.**
- Suppose we create a savepoint having the same name as an previous savepoint, then the previous savepoint is deleted.
  - Once you have created savepoint after that you can perform other functions such as commit, roll back the entire transaction, or roll back to the savepoint.

Transaction Control Language(TCL)

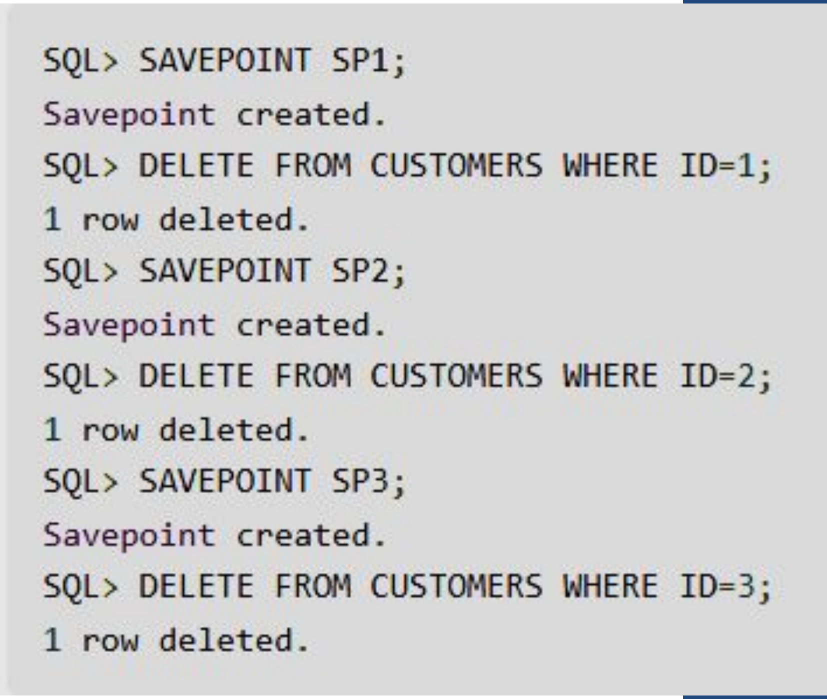
Syntax for a SAVEPOINT command:

- **SAVEPOINT SAVEPOINT_NAME;**
- This command serves only in the creation of a SAVEPOINT among all the transactional statements.



Consider the CUSTOMERS table having the following records.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

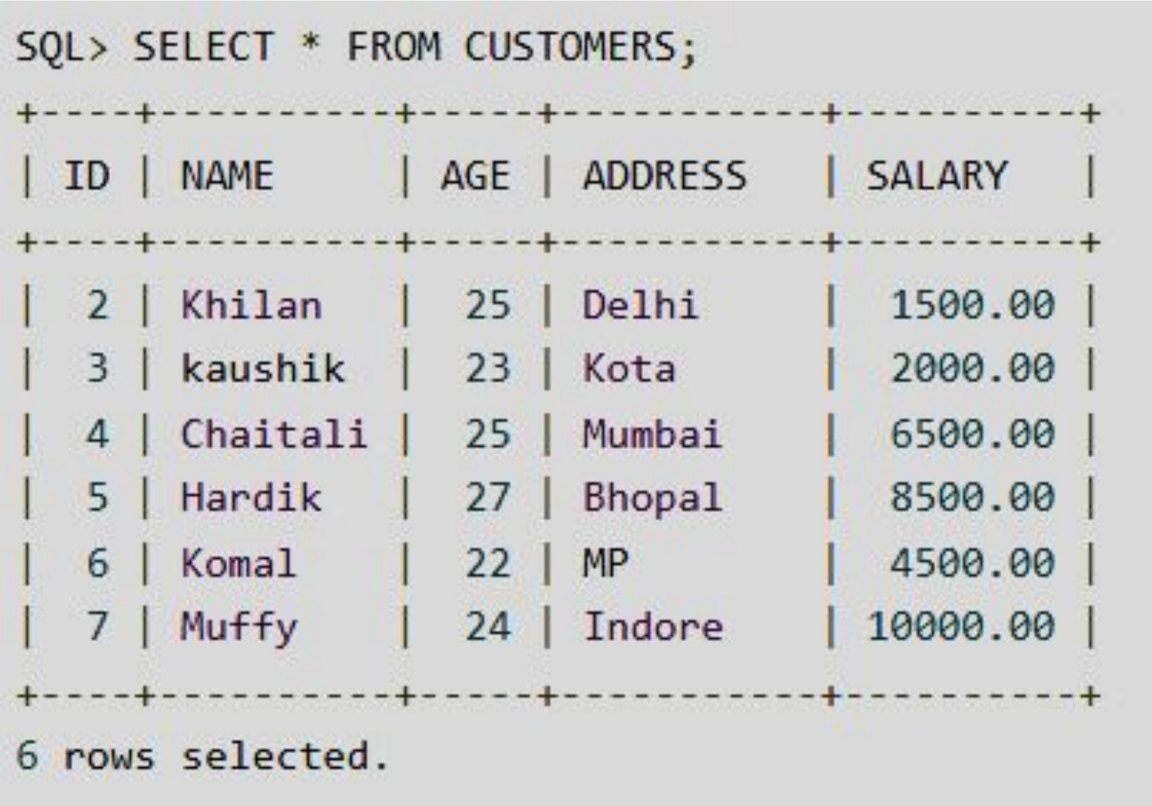


```
SQL> SAVEPOINT SP1;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=1;
1 row deleted.
SQL> SAVEPOINT SP2;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=2;
1 row deleted.
SQL> SAVEPOINT SP3;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=3;
1 row deleted.
```

Transaction Control Language(TCL)

mysql> **SAVEPOINT SP1;**

mysql> **SELECT * FROM CUSTOMERS;**



```
SQL> SELECT * FROM CUSTOMERS;
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
6 rows selected.
```

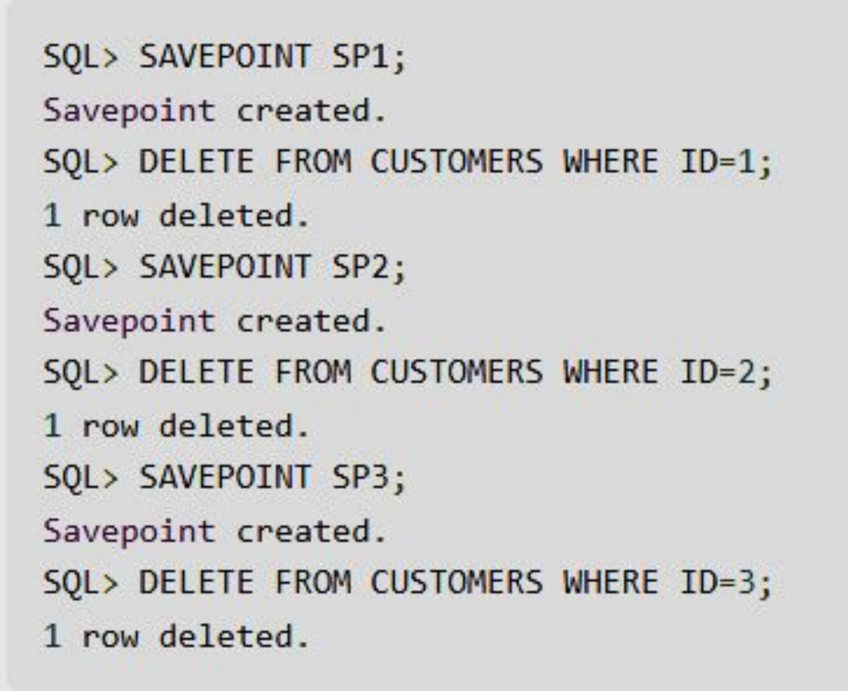
TCL(Transaction Control Language)

ROLLBACK Command

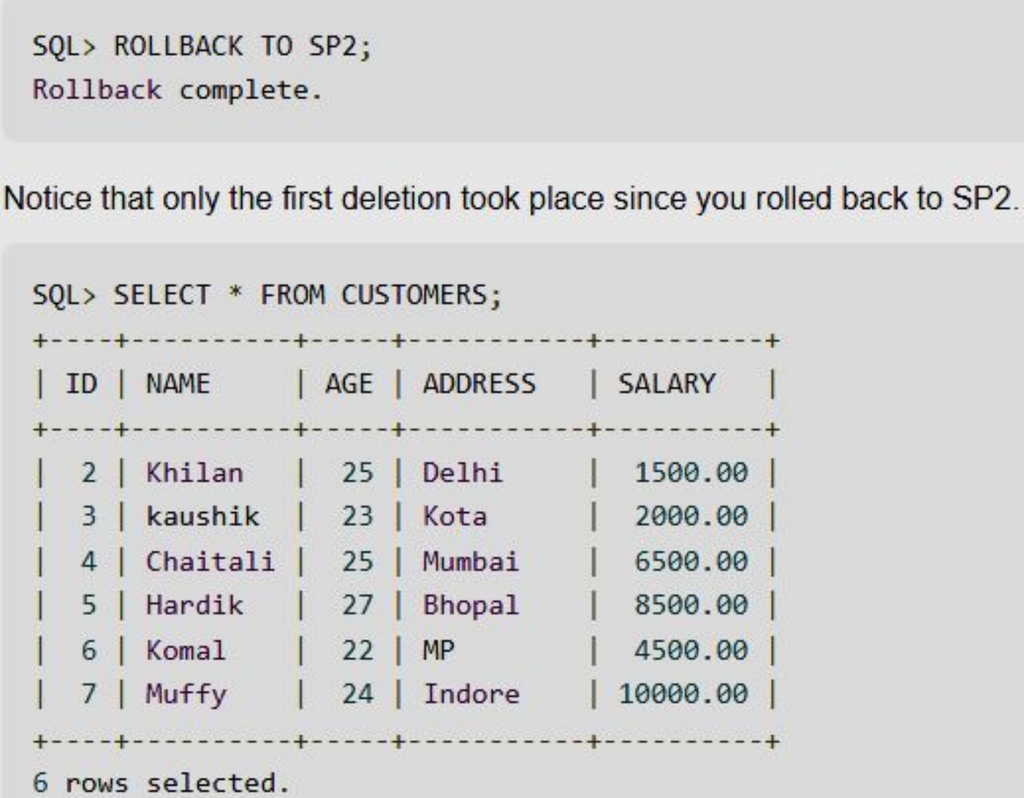
- The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.
- This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.
- The **syntax** for a ROLLBACK command is as follows −

```
ROLLBACK TO SAVEPOINT_NAME ;
```

TCL(Transaction Control Language)



```
SQL> SAVEPOINT SP1;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=1;
1 row deleted.
SQL> SAVEPOINT SP2;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=2;
1 row deleted.
SQL> SAVEPOINT SP3;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=3;
1 row deleted.
```



```
SQL> ROLLBACK TO SP2;
Rollback complete.
```

Notice that only the first deletion took place since you rolled back to SP2.

```
SQL> SELECT * FROM CUSTOMERS;
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
6 rows selected.
```

TCL(Transaction Control Language)

COMMIT Command

- Changes made to the database by INSERT, UPDATE and DELETE commands are temporary until explicitly committed
- On execution of this command all changes to the database made by you are made permanent and cannot be undone
- **Syntax**: COMMIT [Work]

Exercise:1

- **For given database, write SQL queries:**

  EMPLOYEE(e_id,name,street,city)

  WORKS(eid,cid,salary)

  MANAGER(eid,manager_name)

  COMPANY(cid,company_name,city)

1. Modify database so that ‘Jack’ NOW LIVES IN ‘New York’
2. Give all employees of ‘ANZ corporation’ a 10% raise in salary

Solution: 1

```
1.  mysql>
    UPDATE
    EMPLOYEE
SET city= ‘New York’
WHERE
name=‘Jack’;


2.  mysql>
    UPDATE
    WORKS
SET salary = (salary+(0.1*salary))
WHERE cid IN(SELECT cid FROM COMPANY
```

Exercise:2

- **For given database, write SQL queries:**

PERSON(driver_id#, name, address)

CAR(license, model, year)

ACCIDENT(report_no, date, location)

OWNS(driver_id#, license)

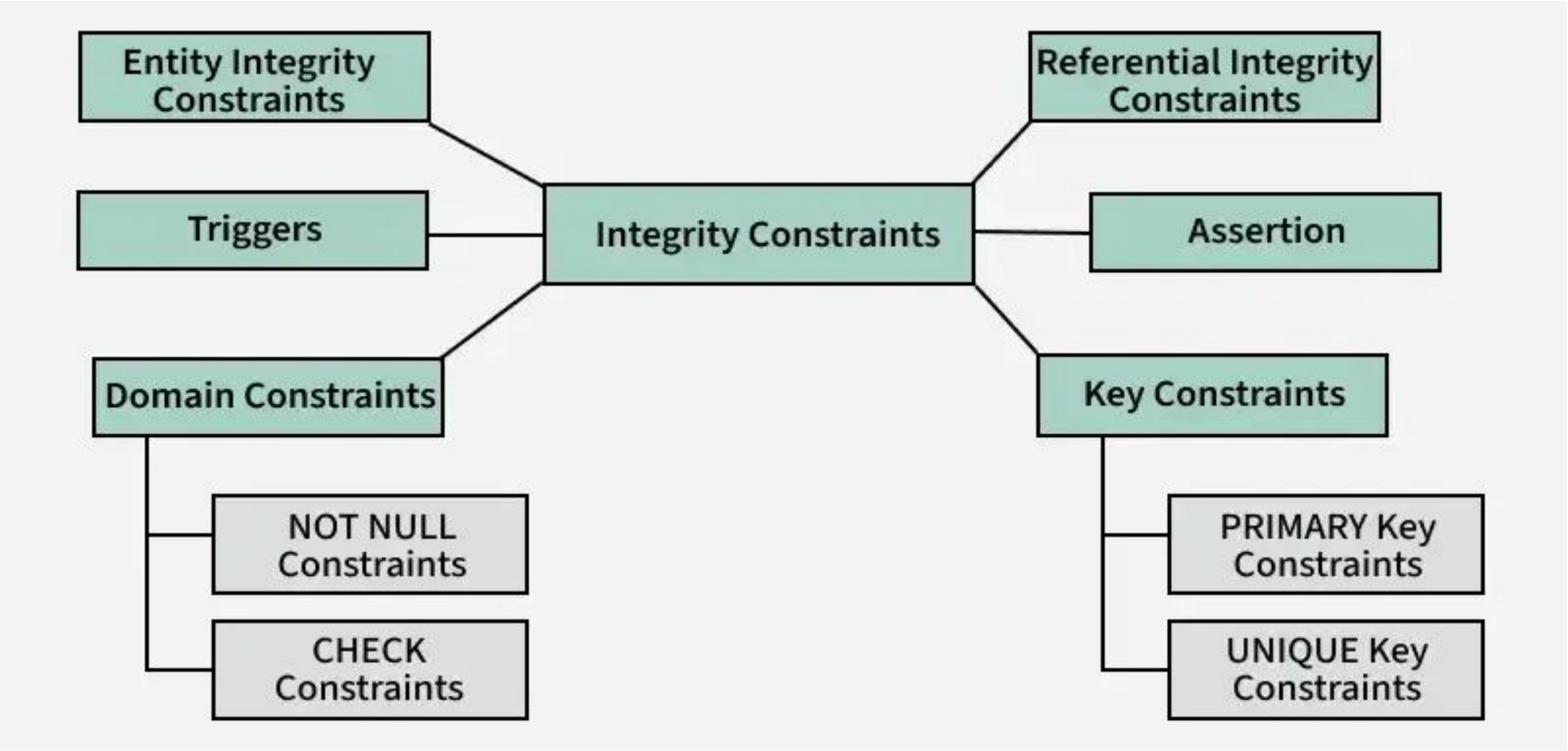
PARTICIPATED (driver_id, car, report_number, damage_amount)

1. Update damage amount for car with license number "Mum2022" in the accident with report number "AR2197" to Rs. 5000

Solution: 2

```
mysql>
UPDATE PARTICIPATED
SET damage_amount = 5000
WHERE report_number LIKE ‘AR2197’ AND car = ‘Mum2022’;
```

Integrity Constraints



Integrity Constraints

- Mainly **security and integrity** of a database is the most important factors in judging the success of system.
- Integrity constraint is a mechanism to prevent invalid data entry into table to maintain the data consistency.
- Constraints are used to enforce limits to the range of data or type of data that can be inserted/updated/deleted from a table
- The whole purpose of constraints is to maintain the data integrity during the various transactions like update/delete/insert on a table.

Types of Constraints

- There are different types of constraints:

1. Domain Integrity Constraints
2. Entity Integrity Constraints
3. Referential Integrity Constraints
4. Enterprise Constraints

Domain Integrity Constraints

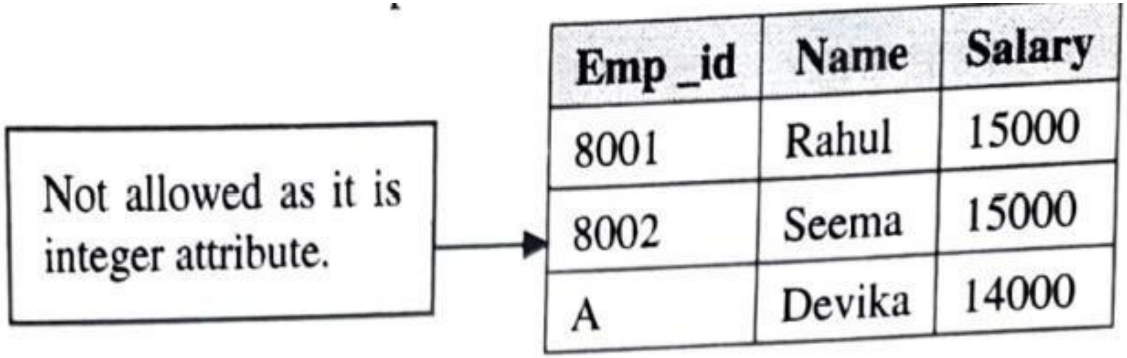
- The domain constraints are considered as the most basic form of integrity constraints.
- For attribute, domain integrity constraint defines the **default value, the range value or specific value.**
- The domain integrity constraints are easy to test when data is entered.
- The domain integrity constraints check that whether the attribute having proper and right value in the database or not.
- Domain integrity means it is the collection of valid set of values for an attribute.

Domain Integrity Constraints

- Constraints:
  - **Not Null**
  - **Unique**
  - **Default**
  - **Check**

Domain Integrity Constraints

- **Data Type**
  - A domain is the set of all unique values which are permitted for an attribute.
  - Domain constraints are user defined data type.
  - As we say that domain is the set of unique values, the column for which domain constraint has set, contains same type of data, based on its data type.
  - The column does not accept values of any other data type



Not allowed as it is integer attribute.

Emp _id	Name	Salary
8001	Rahul	15000
8002	Seema	15000
A	Devika	14000

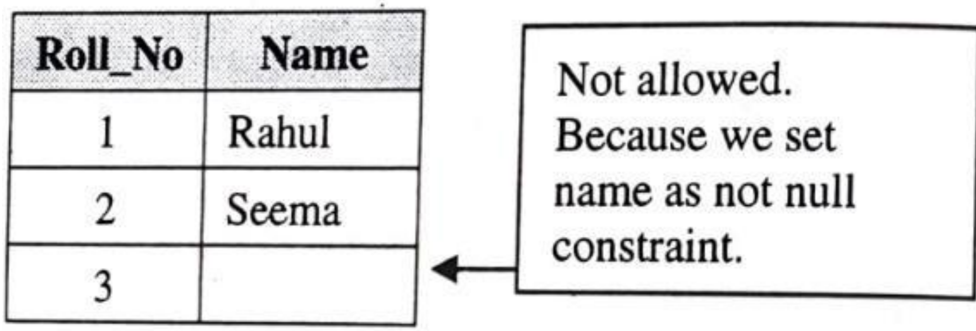
NOT NULL Constraints

- By setting the NOT NULL constraint we can assure that a column does not hold a NULL value
- When for a specific column, no value is provided while inserting a record into a table, by default it takes NULL values
- Example: Consider table student having 'name' field with NOT
  NULL constraint.
- **Syntax**:

CREATE TABLE Student
(Roll_No int NOT NULL,

Name varchar(10) NOT

NULL



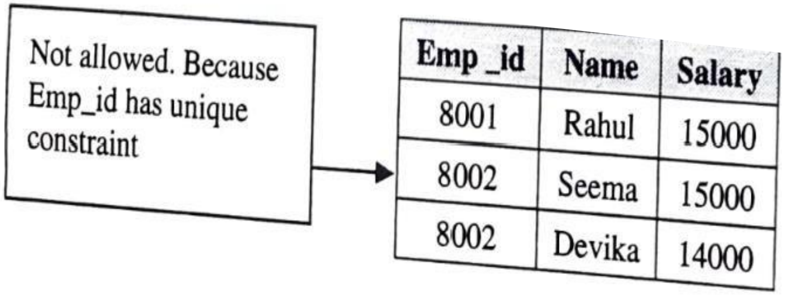
Roll_No	Name
1	Rahul
2	Seema
3	

Not allowed.
Because we set
name as not null
constraint.

Unique Constraint

- UNIQUE Constraint as the name suggests, it can take only unique values in a column or set of columns
- It keeps uniqueness of the table.
- When a column has a unique constraint then that particular column cannot have duplicate values in it.
- **Syntax**:

```
CREATE TABLE Employee
(Emp_id int UNIQUE,
Name varchar(10),
Salary Double(10,2)
);
```



Not allowed. Because Emp_id has unique constraint

Emp _id	Name	Salary
8001	Rahul	15000
8002	Seema	15000
8002	Devika	14000

Default Constraint

- When a user does not provide a value to the column while inserting the records in the table, the DEFAULT constraint provides a default value to that column.
- Example: We can set DEFAULT Constraint by assigning the value 10000 to the column exam_fees in student table
- **Syntax**:

```
CREATE TABLE Student(
Roll_No int NOT NULL,
Name varchar (25),
Fees int DEFAULT 10000
);
```

Check Constraint

- This constraint is used to set user defined constraint for the column
- As per the requirements of business for which we are developing the application, we may have to set some rules while inserting or updating data on specific field/ attribute
- **Syntax:**

```
CREATE TABLE Student(

Roll_No int UNIQUE,
Name varchar(25),
Age int,
CHECK Age between 15 and 20);
```

Entity Integrity Constraints

Key Constraint – Primary Key

- Under Entity Integrity Constraint Primary key is the main factor
- Primary key uniquely identify each record in a table.
- It must have unique values and cannot hold null values
- Primary key is the combination of NOT NULL & UNIQUE constraints.

Not allowed as primary key cannot be null

Emp_id	Name	Salary
8001	Rahul	15000
8002	Seema	15000
	Devika	14000

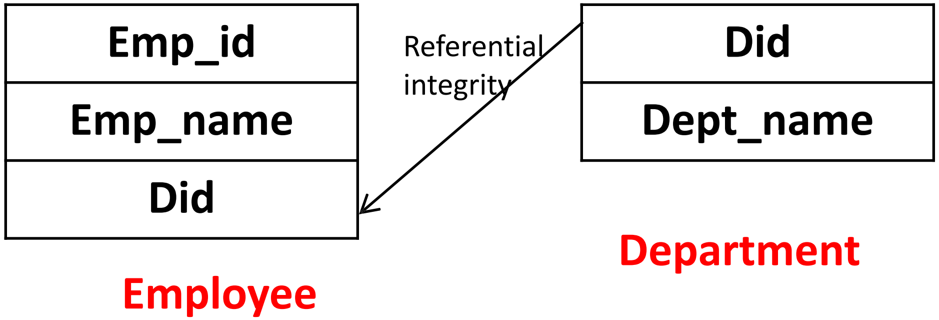
Entity Integrity Constraints

- Here we set primary key to emp_id table. If we add any repeated value or null value in the column it will because primary key never contains null or repeated values.
- **Syntax:**

```
CREATE TABLE Employee(

Emp_id int,

Name varchar(25),
Salary char(4),
PRIMARY KEY (Emp_id)
)
```

Referential Integrity Constraints

- A value appearing in a one relation (table) for a given set of attributes also appears for another set of attributes in another relation (table). This is called **referential integrity.**
- The referential integrity constraint is specified between two tables to maintain the consistency among tuples in the two tables.
- The tuple in one relation refers only to an existing tuple in another relation.



Emp_id

Emp_name

Did

Employee

Referential integrity

Did

Dept_name

Department

Referential Integrity Constraints

Emp_id	Emp_name	Did
1	Sam	20
2	Suhas	10
3	Jay	20
4	Om	10

Did	Dept_name
10	HR
20	TIS
30	L&D

- Ex: Employee table has Did as foreign key reference to **Did** column in Department table this is called as referential integrity.
- Here we are forcing the database to check the value of Did column from the department table while inserting any value in Employee table.
- This helps to maintain data consistency

Referential Integrity Constraints

Foreign key violations in SQL

- If any row in EMP table is added with "Did” value which is not there in department table the insert statement will give foreign key violation error.
- In previous example, we will refer Department as parent table (as it is containing Primary key) and Employee table as Child table (as it is containing Foreign Key).
- There are 4 problems which causes the foreign key violations:

A. Adding new tuple to Child Table (Add Child)

B. Updating tuple from Child Table

C. Deleting tuple from Parent Table

D. Updating tuple from Parent Table

Referential Integrity Constraints

a) Adding new tuple to Child Table (Add Child)

- If we try to add an employee with Did 70 to employee table(child table) , it will return foreign key violation error
- As Did 70 is not there in Department table(Parent table)

Example:

- INSERT INTO Employee VALUES (11,'Devid', 70);

Output

- ORA-02291: Integrity constraint (Employee.FK_Employee) violated - parent key not found

Emp_Id	**Emp_name**	**Did**
11	Devid	70

- This functionality helps to maintain data consistency in database.

Referential Integrity Constraints

b) Updating tuple from Child Table

- If we try to update an employee Emp_id = 2 with Did as 70 to employee table (Child Table), it will return foreign key violation error.
- As Did 70 is not there in Department table (Parent table)
- **Example**:

```
UPDATE Employee
SET Did= 70 WHERE Emp_id=2;
```

Output:

```
ORA-02291 Integrity constraint (Employee.FK_Employee) violated - parent key not found
```

- This functionality helps to maintain data consistency in database.

Referential Integrity Constraints

c) Deleting tuple from Parent Table

- If we try to delete department Did = 10 from Department table (Parent table), it will return foreign violation error.
- As there are few employees working in department with Did =10.
- **Example:**

DELETE Department WHERE Did=10;

- *Output:*

ORA-02292: integrity constraint(Employee.FK_Employee) violated - child record found.

- This functionality will creates limitation for deletion of parent record if it has some associated child records

Referential Integrity Constraints

d) Updating tuple from Parent Table

- If we try to update department of Did = 10 with Did = 70, it will return foreign key violation are few employees still working in department with Did = 10.
- **Example:**

UPDATE Department SET Did = 70 WHERE Did = 10;

Output:

ORA-02292: integrity constraint (Employee.FK_Employee) violated - child record found.

- This functionality will creates limitation for updating parent
  record if it has some associated child records

Referential Integrity Constraints

Delete-Update (DU) rules to solve problem of foreign key violation

- If any row in EMP table is added with 'Did' value which is not there in department table then insert statement will give foreign key violation error
- This rule can be enforced as given as

  follows: Create Table Employee(

Eid varchar (50) Primary Key,

. . .

Did varchar (50) foreign key references department

(Did) On delete CASCADE

On update CASCADE);

Referential Integrity Constraints

NO ACTION / RESTRICT

- This clause will discards the delete or update operation on the parent table
- In this case the database engine will not allow user to delete the row and using FK violation error.
- The RESTRICT rule will not allow you to delete a row from the parent table although as there corresponding row present in child table.

```
Create Table Employee(
Eid varchar (50) Primary Key,
…
Did varchar (50) foreign key references
department (Did)
On delete RESTRICT
On update RESTRICT) ;
```

Referential Integrity Constraints

- The database engine will give the error and the delete action on the row in the parent table is ignored
- Deletion of department is not allowed as there is some employees are present in that department

CASCADE

- Corresponding rows are deleted from the referencing table (Child table), if that row is deleted from parent table.
- If a department is deleted then all the employee records that refers to the deleted department are also been deleted

```
Create Table Employee(
Eid varchar (50) Primary Key,
…
Did varchar (50) foreign key references
department (Did)
On delete CASCADE
On update CACADE) ;
```

Referential Integrity Constraints

SET NULL

- Foreign Key data value is set to NULL, if the corresponding row in the parent table is deleted
- For this constraint to execute, the foreign key columns must be nullable
- Insert Null value of did in the place of deleted did in employee table.

```
Create Table Employee(
Eid varchar (50) Primary Key,
…
Did varchar (50) foreign key references department
(Did)
On delete SET NULL
On update SET NULL) ;
```

Referential Integrity Constraints

SET DEFAULT

- Foreign key data values refer to non-existing foreign key are set to their default values.
- For this constraint to execute, all foreign key columns must have default definitions.
- If a column is null able, and there is no explicit default value set, NULL becomes the implicit default value of the column.
- Insert any default value of 'did' (which exists in the departing table) in the place of deleted 'Did'

```
Create Table Employee(
Eid varchar (50) Primary Key,
…
Did varchar (50) foreign key references department
(Did)
On delete SET DEFAULT
On update SET DEFAULT) ;
```

Difference: Primary key and Foreign Key Constraint

Parameter	Primary Key	Foreign Key
Function	Primary key uniquely identify a record in the table.	Foreign key is a field in the table that is primary key in another table.
Null	Primary Key can't accept null values.	Foreign key can accept null values.
Index	By default, Primary key is clustered index and data in Foreign key do not automatically create an index, the database table is physically organized in the clustered or non-clustered. You can manually sequence of clustered index.	Foreign key do not automatically create an index, clustered or non-clustered. You can manually create an index on foreign key
Number	Only one primary key in a table	More than one foreign key in a table

Referential Integrity Constraints

Enterprise Constraints

- Enterprise Constraints are also referred as Semantic constraints.
- They are additional rules specified by users or database administrators.
- These **rules are depending upon the requirements and constraints of the business** for which the database system is being maintained.
- For Example:
  - In College System a class can have a maximum of 30 students. A teacher can teach a maximum of 4 classes a semester.
  - In Corporate System an employee cannot take a part in more than 5 projects. Salary of an employee cannot exceed salary of the employee's manager

Thank You!!