DATA DEFINITION LANGUAGE (DDL)

--To create a database with a proper character set and collation

CREATE

1) CREATE DATABASE

```
CREATE DATABASE IF NOT EXISTS profile_maker CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
```

2) CREATE TABLE

```
--Creating a `users` table
CREATE TABLE `users` (
  `id` int(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `username` varchar(25) NOT NULL UNIQUE,
  `password` varchar(25) NOT NULL,
  `prefix` enum('Mr.','Mrs.','Ms.', 'Mx.') DEFAULT NULL,
  `name` varchar(25) DEFAULT NULL,
  `email` varchar(50) DEFAULT NULL,
  `mobile` varchar(10) DEFAULT NULL,
  `age` tinyint UNSIGNED DEFAULT NULL,
  `gender` enum('Male','Female','Genderqueer', 'Undisclosed') DEFAULT NULL,
  `state` varchar(30) DEFAULT NULL,
  `profilePic` varchar(50) DEFAULT NULL,
  `resume` varchar(50) DEFAULT NULL,
  `creationTime` datetime DEFAULT CURRENT_TIMESTAMP,
  `modificationTime` datetime DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  CONSTRAINT `PK_users` PRIMARY KEY (`id`),
  CONSTRAINT `UQ_users_email` UNIQUE (`email`),
  CONSTRAINT `UQ_users_profilePic` UNIQUE (`profilePic`),
  CONSTRAINT `UQ_users_resume` UNIQUE (`resume`),
  CONSTRAINT `CHK_users_mobile` CHECK(`mobile` is null or `mobile` regexp
'^[0-9]{10}$'),
  INDEX `IX_users_email` (`email`),
  INDEX `IX_users_username_password` (`username`, `password`)
) ENGINE=InnoDB;
```

3) Creating Table from Existing Table

We can take structure only, take structure plus data.

```
sql> create table hm1 as select * from dept;
```

4) **CREATING TEMPORARY TABLE**

```
CREATE TEMPORARY TABLE new_tbl SELECT * FROM orig_tbl LIMIT 0;
```

ALTER

1) **Adding Primary Key Constraint (Table Level)**

```
ALTER TABLE `job_history` ADD CONSTRAINT `PK_jobhistory` PRIMARY
KEY(`employee_id`);
```

2) **Adding Unique Key Constraint (Table Level)**

```
ALTER TABLE `job_history` ADD CONSTRAINT `UQ_jobhistory`
UNIQUE(`employee_id`);
```

3) **Adding Foreign Key COnstraint**

```
ALTER TABLE `job_history` ADD CONSTRAINT `FK_jobs_jobhistory` FOREIGN
KEY(`job_id`) REFERENCES `jobs`(`job_id`) ON DELETE CASCADE;
                        Or
      For Multiple Foreign Keys From Same Table
ALTER TABLE `employees` ADD CONSTRAINT `FK_departments_employees`
FOREIGN KEY(`manager_id`,`department_id`)
REFERENCES `departments`(`manager_id`, `department_id`);
```

4) **Add a Column to Table**

```
ALTER TABLE <TABLE NAME>
ADD <COLUMN NAME> <DATATYPE>;
```

5) **Remove a Column From a Table**

```
ALTER TABLE <TABLE NAME>
DROP <COLUMN NAME>;
```

6) **Modify a Column**

```
ALTER TABLE <TABLE NAME>
MODIFY COLUMN <COLUMN NAME> <DATATYPE>;
```

TRUNCATE

TRUNCATE TABLE empties a table completely. It requires the DROP privilege. Logically, TRUNCATE TABLE is similar to a DELETE statement that deletes all rows, or a sequence of DROP TABLE and CREATE TABLE statements.

To achieve high performance, TRUNCATE TABLE bypasses the DML method of deleting data. Thus, it does not cause ON DELETE triggers to fire, it cannot be performed for InnoDB tables with parent-child foreign key relationships, and it cannot be rolled back like a DML operation. However, TRUNCATE TABLE operations on tables that use an atomic DDL-supported storage engine are either fully committed or rolled back if the server halts during their operation.

Syntax

TRUNCATE TABLE table_name;

DROP

1. Delete a Database
   DROP DATABASE dbName;
2. Delete/Drop a Column
   ALTER TABLE table_name
   DROP [COLUMN] column_name;

Referential Actions

In SQL, a DELETE constraint specifies the action to be taken when a referenced record is deleted in a table with a foreign key relationship.

1. **CASCADE**: Deletes all records in the referencing table (child table) when the referenced record (parent table) is deleted. Both ON DELETE CASCADE and ON UPDATE CASCADE are supported.
   Note : Cascaded foreign key actions do not activate triggers.
2. **SET NULL**: Sets the foreign key values in the referencing table to NULL when the referenced record is deleted.
3. **SET DEFAULT**: Sets the foreign key values in the referencing table to the default value when the referenced record is deleted.This action is recognized by the MySQL parser, but both **InnoDB** and **NDB** reject table definitions containing ON DELETE SET DEFAULT or ON UPDATE SET DEFAULT clauses.
4. **RESTRICT**: Prevents deletion of the referenced record if it is referenced by a record in the referencing table. RESTRICT (or NO ACTION) is the same as omitting the ON DELETE or ON UPDATE clause.

Example syntax:

```
ALTER TABLE child_table
ADD CONSTRAINT fk_name
FOREIGN KEY (column_name)
REFERENCES parent_table (column_name)
ON DELETE CASCADE;
```

CHECK CONSTRAINT IN SQL

A `CHECK` constraint is specified as either a table constraint or column constraint:

- A table constraint does not appear within a column definition and can refer to any table column or columns.
- A column constraint appears within a column definition and can refer only to that column.

Consider this table definition:

```
CREATE TABLE t1
( CHECK (c1 <> c2),
  c1 INT CHECK (c1 > 10),
  c2 INT CONSTRAINT c2_positive CHECK (c2 > 0),
  c3 INT CHECK (c3 < 100),
  CONSTRAINT c1_nonzero CHECK (c1 <> 0),
  CHECK (c1 > c3));
```

The definition includes table constraints and column constraints, in named and unnamed formats:

- The first constraint is a table constraint: It occurs outside any column definition, so it can (and does) refer to multiple table columns. This constraint contains forward references to columns not defined yet. No constraint name is specified, so MySQL generates a name.
- The next three constraints are column constraints: Each occurs within a column definition, and thus can refer only to the column being defined. One of the constraints is named explicitly. MySQL generates a name for each of the other two.
- The last two constraints are table constraints. One of them is named explicitly. MySQL generates a name for the other one.