SQL: DDL, DML and DTL

- Data Definition Language (DDL) : To create or modify the database structure. CREATE, ALTER, DROP, TRUNCATE
- Data Manipulation Language (DML) : To add, modify and delete records. INSERT, UPDATE, DELETE.
- Data Transaction Language (DTL) : To manage transactions. START TRANSACTION, COMMIT, ROLLBACK

Objective: Objective of this project is to create database, schema, tables and columns; add and drop columns; drop schemas and tables; insert, update and delete records; create indexes; add unique constraints; change column to not null; implement stored procedures and triggers.

1) Creating schema and tables

```
CREATE SCHEMA marketing_project DEFAULT CHARACTER SET utf8mb4;
USE marketing_project;
CREATE TABLE marketing_campaign (
    campaign_id INT NOT NULL,
    campaign_name VARCHAR(65),
    created_at TIMESTAMP,
    PRIMARY KEY (publisher_id)
);
CREATE TABLE campaign_spend (
    campaign_spend_id VARCHAR(45) NOT NULL,
    campaign_id INT NOT NULL,
    month DATE NOT NULL,
    spend DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (campaign_spend_id)
);
```

2) Drop, add and modify columns

```
USE new_schema;

ALTER TABLE employees
DROP COLUMN over_time;

ALTER TABLE employees
ADD COLUMN avg_customer_rating DECIMAL(10,1) AFTER created_at;

ALTER TABLE employees
MODIFY COLUMN avg_customer_rating DECIMAL(10,2) NOT NULL;

ALTER TABLE customers
ADD PRIMARY KEY (customer_id);
```

3) Drop schema and table

```
USE sales_div01;
DROP SCHEMA sales_div01;

USE sales_div02;
DROP TABLE old_msrp;
```

4) Insert, update and delete records

```
USE supplychain;
-- if the new data has no missing values
INSERT INTO inventory VALUES
(11,product1, 2),
(12,product2, 3);

-- if missing value in any column
INSERT INTO inventory (product_ID, product_name) VALUES
(14,product5);

-- UPDATING DATA (use with WHERE clause) --
USE supplychain;
-- Primary key is must for WHERE condition
UPDATE inventory
SET number_in_stock = 0
WHERE inventory_id IN (1,9); -- always run UPDATE with primary key --

-- DELETING RECORDS/ROWS --
SELECT @@autocommit;
-- if planning to rollback deleted record or row --
SET autocommit = 0 ;
START TRANSACTION;
DELETE FROM inventory
WHERE inventory_id = 7; -- always run DELETE with primary key and WHERE --
DELETE FROM customers
WHERE customer_id BETWEEN 1 AND 6;
-- retrieve deleted record or rows once deleted. only works when autocommit is OFF
ROLLBACK;
-- permanent change --
COMMIT; -- If commit is written at the end, ROLLBACK won't work even autocommit is set to 0.

-- Cannot be rolled back.
-- Records will be permanently deleted from table even if autocommit is set to 0.
TRUNCATE TABLE customers;
```

5) Index, unique, procedure and trigger

```
-- ADDING INDEX (to improve query perfromance) --
ALTER TABLE customer_reviews
ADD INDEX employee_id (employee_id ASC) VISIBLE;
-- OR can write
CREATE INDEX employee_id ON customer_reviews (employee_id ASC) VISIBLE;
-- DROPPING INDEX --
ALTER TABLE customer_reviews
DROP INDEX employee_id;

-- ADDING UNIQUE (allow only distinct value in each row) --
ALTER TABLE inventory
ADD UNIQUE (inventory_id);
-- OR
ALTER TABLE inventory
ADD CONSTRAINT u_inventory_id UNIQUE (inventory_id);
-- DROPPING UNIQUE CONSTRAINTS --
ALTER TABLE inventory
DROP INDEX u_inventory_id;
```

Created by: Mustafa Rangwala

```sql
-- NOT NULL CONSTRAINT --
ALTER TABLE inventory
CHANGE COLUMN number_in_stock
number_in_stock BIGINT(20) NOT NULL; -- change column name as well
-- OR
ALTER TABLE inventory
MODIFY COLUMN number_in_stock BIGINT(20) NOT NULL;
-- REMOVE NOT NULL --
ALTER TABLE inventory
MODIFY COLUMN number_in_stock BIGINT(20);

 -- STORED PROCEDURES (store and call frequently used queries) --
 -- changing the delimiter
 DELIMITER //
 -- creating the procedure
 CREATE PROCEDURE Store_procedure_selectalldata()
 BEGIN
     SELECT * FROM data;
 END //
 -- changing the delimiter back to default
 DELIMITER ;

 -- calling the procedure that we have created
 call Store_procedure_selectalldata();

-- Triggers (to automate system task) --
-- Substracting
CREATE TRIGGER PurchaseUpdateInventory
AFTER INSERT ON customer_purchases
FOR EACH ROW
    UPDATE inventory
       -- substracting an item for eachpurchase
        SET number_in_stock = number_in_stock - 1
    WHERE inventory.inventory_id = NEW.inventory_id;

-- Checking if it works
    INSERT INTO customer_purchases VALUES
    (25,0,8), -- inventory_id = 8
    (26,0,9); -- inventory_id = 9

-- Can drop
DROP PROCEDURE IF EXISTS Store_procedure_selectalldata ;
DROP TRIGGER IF EXISTS PurchaseUpdateInventory;
```