S32: SQL DML COMMANDS

DATA MANIPULATION LANGUAGE (DML)?

- **"DML"** stands for **Data Manipulation Language** in **SQL.**
- It's a subset of SQL statements used to define the structure of a database.

INSERT INTO STATEMENT

- The **INSERT INTO** statement in **SQL** is used to add new records to a table in a database.
- There are two main ways to use the **INSERT INTO** statement:
  specifying the columns explicitly and inserting data for all columns without specifying them.
- **Syntax:**

```
INSERT INTO table_name (col1, col2....) VALUES (value1, value2...);
```

- **Example:** Creating the **student's** table:

```
CREATE TABLE college.students(
  stu_id INTEGER PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL
);
```

1. Specifying the columns explicitly:

```
INSERT INTO college.students(user_id, name, email, password)
VALUES (NULL, 'Akash', 'akash@gmail.com', '@1#$%@');
```

stu_id	name	email	password
1	akash	akash@gmail.com	1@#32

2.Inserting data for all columns without specifying:

```
INSERT INTO college.students
VALUES (NULL, 'Goku', 'goku@gmail.com', '24$%@');
```

stu_id	name	email	password
1	akash	akash@gmail.com	1@#32
2	goku	goku@gmail.com	24$%@

2.Inserting multiple data:

```
INSERT INTO college.students
VALUES (NULL, 'max', 'max@gmail.com', '54444*'),
VALUES (NULL, 'virat', 'virat@gmail.com', '22222')
);
```

stu_id	name	email	password
1	akash	akash@gmail.com	1@#32
2	goku	goku@gmail.com	24$%@
3	max	max@gmail.com	54444*
4	virat	virat@gamil.com	22222

STEPS TO IMPORT TABLE DATA INTO MYSQL DATABASE

- **Open MySQL Workbench:**
  Launch MySQL Workbench and connect to your **MySQL** server.
- **Select Your Database:**
  From the left sidebar (Schema), select the database where you want to import the data.
- **Open Table Data Import Wizard:**
  Right-click on the target table and select **"Table Data Import Wizard"**.
- **Select the Data File:**
  Click the **"Browse"** button and select your **CSV/SQL** file.
- **Select Destination:**
  Select the destination and additional options like creating table name etc.
- **Configure Import Settings:**
  On the next screen, you can configure import settings such as Encoding, columns and you can change it's datatypes.
  Ensure the **"First Row Contains Column Names"** option is checked if your **CSV/SQL** file includes headers.

- **Import Data:**
  Click **"Next"** and then **"Next"** again to start the import process. **MySQL** Workbench will display progress and notify you upon completion.
- **Import Results:**
  'Table was created' message will show on page and click on **Finish.**

SELECT STATEMENT

- The **SELECT** statement in **MySQL** is used to query the database and retrieve data.
- This statement is fundamental for accessing and manipulating data in a **MySQL** database.
- Here are examples of how to use the **SELECT** statement effectively.
- **Syntax:**

```
SELECT * FROM database_name.table_name;
```

- Here are examples of how to use the **SELECT** statement effectively.

Select all the rows and all the columns:

```
SELECT * FROM  flipkart.smartphones;
```

Selecting the specific columns:

```
SELECT brand_name, model, os FROM flipkart.smartphones;
```

brand_name	model	os
oneplus	OnePlus 11 5G	android
samsung	Samsung Galaxy A14 5G	android

Using Aliases (AS):
You can rename the column name to make column names more readable:

```
SELECT brand_name AS 'brands', model, os AS 'operating system'
FROM flipkart.smartphones;
```

brands	model	operating system
oneplus	OnePlus 11 5G	android
samsung	Samsung Galaxy A14 5G	android

Creating the mathematical expression from columns:
PPI (Pixels Per Inch) measures the pixel density of a screen or image, indicating how many pixels fit in a 1-inch line. Higher PPI values result in sharper and more detailed visuals.
To calculate PPI (Pixels Per Inch):
Calculate the diagonal pixel count (d_o):

$$d_o = \sqrt{w^2 + h^2}$$

where, **w** is the number of pixels along the horizontal line.
h is the number of pixels along the vertical line.
Calculate the PPI:

$$PPI = d_i / d_o$$

where, d_i is the diagonal screen size in inches.

```
SELECT model,
SQRT(resolution_width*resolution_width + resolution_height*resolution_height)/screen_size AS 'ppi'
FROM flipkart.smartphones;
```

model	ppi
OnePlus 11 5G	525.9210168998101
Samsung Galaxy A14 5G	401.0247511261829

CONSTANT VALUE

- **Adding constant value column for smartphone:**

```
SELECT model, 'smartphone' AS 'type' FROM flipkart.smartphones;
```

model	type
OnePlus 11 5G	smartphone
OnePlus Nord CE 2 Lite 5G	smartphone
Samsung Galaxy A14 5G	smartphone

DISTINCT CLAUSE

- The **DISTINCT** clause in **MySQL** is used to **remove duplicate rows** from the result set of a query.
- When you use **DISTINCT**, **MySQL** ensures that each row in the result set is unique based on the columns you specify.

- **Syntax:**

```
SELECT DISTINCT col1, col2,...
FROM table_name
```

- **Examples:**

Creating a new column with unique brands:

```
SELECT DISTINCT brand_name AS 'all brands'
FROM flipkart.smartphones;
```

all brands
oneplus
samsung
motorola
realme

Find the unique brands in processor_brands columns:

```
SELECT DISTINCT processor_brand AS 'all processors'
FROM flipkart.smartphones;
```

all processors
snapdragon
exynos
dimensity
bionic

Find the unique rows with combination of brand_name & processor_brands columns:

```
SELECT DISTINCT brand_name, processor_brand FROM flipkart.smartphones;
```

brand_name	processor_brand
oneplus	snapdragon
samsung	exynos
motorola	snapdragon
realme	dimensity
samsung	snapdragon

WHERE CLAUSE

- The **WHERE** clause in **MySQL** is used to filter records in a query.
- It specifies the conditions that must be met for the rows to be included in the result set.
- **Syntax:**

```
SELECT col1, col2,...
FROM table_name
WHERE condition;
```

- The **WHERE** clause uses several common operators to filter data based on specific conditions.
- These operators can be grouped into comparison operators, logical operators, pattern matching operators, range checking operators, and set membership operators. Here are the most commonly used operators:
- **Comparison operators:**

Operators	Description
=	Equal to
<> or !=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

- **Logical operators:**

Operators	Description
AND	All conditions must be **TRUE**
OR	At least one condition must be **TRUE**
NOT	Negates a condition

- **Range Checking Operators:**

Operators	Description
BETWEEN	Used to filter results within a specific range (inclusive) **Syntax:** `SELECT col1, col2,... FROM table_name BETWEEN low AND high`

Ex1: Find all phones in the price range of 10000 and 20000

```
SELECT brand_name, model, price FROM flipkart.smartphones
WHERE price BETWEEN 10000 AND 20000;
```

brand_name	model	price
OnePlus	OnePlus Nord CE 2 Lite 5G	19989
samsung	Samsung Galaxy A14 5G	16699
motorola	Motorola Moto G62 5G	14999
realme	Realme 10 Pro Plus	18999

Ex2: (Using multiple conditions)

Find phone with rating greater than 80, price less than 15000 and processor brand must be snapdragon

```
SELECT brand_name, model, rating, price, processor_brand FROM flipkart.smartphones
WHERE price < 15000 AND rating > 80 AND processor_brand = 'snapdragon';
```

brand_name	model	rating	price	processor_brand
motorola	Motorola Moto G62 5G	81	14999	snapdragon
realme	Realme 9 4G	81	14999	snapdragon

- **Set Membership Operators:**

Operators	Description
IN	Checks if a value exists in a set
NOT IN	Checks if a value does not exist in a set

Example: (Using IN)

Find phone that either contains processor brand like snapdragon, exynos or bionic

```
SELECT brand_name, model, processor_brand
FROM flipkart.smartphones
WHERE processor_brand IN ('snapdragon','exynos','bionic');
```

brand_name	model	processor_brand
oneplus	OnePlus 11 5G	snapdragon
motorola	OnePlus Nord CE 2 Lite 5G	snapdragon
samsung	Samsung Galaxy A14 5G	exynos
motorola	Motorola Moto G62 5G	snapdragon
samsung	Samsung Galaxy FE S23 5G	snapdragon
apple	Apple iPhone 14	bionic

Example: (Using NOT IN)

Find phone that not contains processor brand like snapdragon, exynos or bionic

```
SELECT brand_name, model, processor_brand
FROM flipkart.smartphones
WHERE processor_brand NOT IN ('snapdragon','exynos','bionic');
```

brand_name	model	processor_brand
realme	Realme 10 Pro Plus	dimensity
xiaomi	Xiaomi Redmi Note 12 Pro Plus	dimensity
oneplus	One Plus Nord 2T 5G	dimensity
oppo	Oppo A78	dimensity
xiaomi	Xiaomi Redmi Note 12 Pro 5G	dimensity
vivo	Vivo Y16	helio

- **Pattern Matching Operator:**

Operators	Description
LIKE	Used to search for a specified pattern in a column. Common wildcards used with **LIKE** include: • **'%'**:Represents zero or more characters. • **'_'** : Represents a single character.
NOT LIKE	Used to exclude rows that match a specified pattern. It is the opposite of the **LIKE** operator and is also case-insensitive in **MySQL**.

- **NULL Checking Operators:**

Operators	Description
IN NULL	Used to check for **NULL** values
IS NOT NULL	Used to check for **non-NULL** values

- **Examples**:

Find all samsung brands:

```
SELECT brand_name, model, price FROM flipkart.smartphones
FROM brand_name = 'samsung';
```

brand_name	model	price
samsung	Samsung Galaxy A14 5G	16499
samsung	Samsung Galaxy S20 FE 5G	31239
samsung	Samsung Galaxy M53 5G	23790
samsung	Samsung Galaxy S21 FE 5G	39999

Find all phones with price less than 30000:

```
SELECT brand_name, model, price FROM flipkart.smartphones
WHERE price <= 30000;
```

brand_name	model	price
OnePlus	OnePlus Nord CE 2 Lite 5G	19989
samsung	Samsung Galaxy A14 5G	16699
motorola	Motorola Moto G62 5G	14999
realme	Realme 10 Pro Plus	24999

Find the unique rows with combination of brand_name & processor_brands columns:

```
SELECT DISTINCT brand_name, processor_brand FROM flipkart.smartphones;
```

brand_name	processor_brand
oneplus	snapdragon
samsung	exynos
motorola	snapdragon
realme	dimensity
samsung	snapdragon

QUERY EXECUTION ORDER IN MYSQL

- Here is the order in which the clauses are used:

FROM >> WHERE >> GROUP BY >> HAVING >> SELECT >> ORDER BY >> LIMIT

- This order ensures efficient and accurate query processing in **MySQL**.
  1. **FROM:**
     Defines the source tables and how they are joined together.
     This step forms the basis of the dataset to be queried.
  2. **WHERE:**
     Filters rows based on specified conditions before any grouping takes place. Only rows that meet these conditions are included in the next steps.
  3. **GROUP BY:**
     Groups rows that have the same values in specified columns into summary rows.
     Often used with aggregate functions like **COUNT, SUM, AVG**.
  4. **HAVING:**
     Filters groups based on a condition applied to aggregate functions, similar to the **WHERE** clause but for groups.
  5. **SELECT:**
     Specifies which columns or expressions to return in the final result set.
     This step occurs after **WHERE**, **GROUP BY**, and **HAVING** to determine the final output.
  6. **DISTINCT:**
     Removes duplicate rows from the result set. Applied after the **SELECT** clause.
  7. **ORDER BY:**
     Sorts the result set by one or more columns. The sorting can be in ascending **(ASC)** or descending **(DESC)** order.
  8. **LIMIT:**
     Restricts the number of rows returned by the query to a specified number.
- **Remembering the order of execution using acronym representation**: **FWGHS-DOL**
- **Remembering the order of execution with mnemonic or structured phrase:**
  "**F**irst **W**ill **G**racefully **H**ave **S**ome **D**elicious **O**rdered **L**emonade"
  OR
  "**F**inding **W**aldo **G**ets **H**im **S**ome **D**elicious **O**range **L**ollipops"

UPDATE STATEMENT

- The **UPDATE** statement in **MySQL** is used to modify existing records in a table.

SET CLAUSE

- The **SET** clause in **MySQL** is used in various **SQL** statements to specify new values for columns in a table.
- It's most commonly associated with the **UPDATE** statement but is also used in **INSERT**, **REPLACE**, and **variable assignment**.
- **Syntax:**

```
UPDATE table_name
SET col1=value1, col1=value2
WHERE condition;
```

- **Example: UPDATE** Statement with **SET** Clause:

Users table:

stu_id	name	email	password
1	akash	akash@gmail.com	1@#32
2	hitman	hitman@gmail.com	24$%@

1. Updating the email:

```
UPDATE college.students
SET email = 'asp@gmail.com'
WHERE  name = 'hitman';
```

stu_id	name	email	password
1	akash	asp@gmail.com	1@#32
2	hitman	hitman@gmail.com	24$%@

2.Update multiple columns:

```
UPDATE college.students
SET email = 'rohit@gmail.com', email = '12345'
WHERE  name = 'hitman';
```

stu_id	name	email	password
1	akash	akash@gmail.com	1@#32
2	hitman	rohit@gmail.com	12345

DELETE STATEMENT

- The **DELETE** statement in **MySQL** is used to remove rows from a table.
- This operation can be specified to affect only certain rows based on a condition or all rows if no condition is specified.
- **Syntax:**

```
DELETE  FROM  table_name
WHERE conditions;
```

Ex1: (Delete based on condition)

Delete the all phones where price greater than 200000

```
DELETE  FROM flipkart.smartphones
WHERE price > 200000;
```

Ex2: (Delete based on multiple conditions)

Delete the all phones where price greater than 200000

```
DELETE FROM flipkart.smartphones
WHERE primary_camera_rear > 150 AND brand_name = 'samsung';
```

- **NOTE:**

By following these precautions, you can perform UPDATE and DELETE operations more safely and effectively.

- ✓ **Backup Data:** Always create a backup before making changes.
- ✓ **Verify with SELECT:** Run **SELECT** queries to preview affected data.
- ✓ **Careful WHERE Clauses:** Always include specific **WHERE** clauses.
- ✓ **Double-Check Syntax:** Review query syntax before execution.

SQL FUNCTIONS

- SQL provides many built-in functions to perform calculations on data.
- Operations such as mathematical calculations, string manipulation like concatenation, substring are performed using these SQL Functions.
- SQL functions can be broadly categorized into two types: **built-in functions** and **user-defined** functions.

BUILT-IN FUNCTIONS

- Built-in functions are provided by SQL databases and are readily available for use.
- These functions perform various operations like calculations, string manipulations, date and time processing, etc.
- Built-in functions divided into the following two categories.
  - ✓ **Aggregate Functions**
  - ✓ **Scalar Functions**

AGGREGATE FUNCTIONS

- Aggregate functions in SQL perform calculations on a set of values and return a single value.

MAX() :
Finds the maximum value in a specified column.

Ex1: Find the maximum price from price column

```
SELECT MAX(price) FROM flipkart.smartphones;
```

MIN(price)
199990

Ex2: Find the price of the costliest samsung phone

```
SELECT MAX(price) FROM flipkart.smartphones;
WHERE brand_name = 'samsung';
```

MAX(price)
163980

MIN():
Finds the minimum value in a specified column.

Ex1: Find the minimum price from price column

```
SELECT MIN(price) FROM flipkart.smartphones;
```

MIN(price)
3499

Ex2: Find the minimum price from price column

```
SELECT MIN(ram_capacity) FROM flipkart.smartphones;
```

MIN(ram_capacity)
1

AVG():
Computes the average of the values in a specified column.

Ex: Find the average rating of the apple phones

```
SELECT AVG(rating) FROM flipkart.smartphones;
WHERE brand_name = 'apple';
```

AVG(rating)
73.45652173913044

SUM():
Adds up the values in a specified column.

Ex: Find the sum of price column

```
SELECT SUM(price) FROM flipkart.smartphones;
```

price
299451126

COUNT():
Counts the number of rows that match a specified condition.

Ex: Find the number of oneplus phones

```
SELECT COUNT(*) FROM flipkart.smartphones;
WHERE brand_name = 'oneplus';
```

COUNT(*)
42

COUNT(DISTINCT):
Counts unique, non-null values in a specified column.

Ex: Find the number of oneplus phones

```
SELECT COUNT(DISTINCT(brand_name))
FROM flipkart.smartphones;
```

COUNT(DISTINCT(brand_name))
45

STD() :
Standard deviation is a measure of the amount of variation or dispersion in a set of values.

Ex1: Finds the standard deviation of screen_size column

```
SELECT STD(screen_size) FROM flipkart.smartphones;
```

STD(screen_size)
0.3448797557536047

VARIANCE():
Variance measures how far the values are spread out from their average value.

Ex1: Finds the variance of screen_size column.

```
SELECT VARIANCE(screen_size) FROM flipkart.smartphones;
```

VARIANCE(screen_size)
0.11894204592866604

SCALAR FUNCTIONS

- Scalar functions in SQL operate on individual values and return a single value.
- They are used to perform operations on data such as calculations, string manipulations, date and time processing, and other transformations.
- Scalar functions take one or more input values and produce a single output value for each row in a query.

ABS():
Useful for ensuring that values are non-negative, such as when calculating distances or differences.

```
SELECT ABS(-15) AS absolute_value;
```

absolute_value
15

ROUND():
Rounds a number to a specified number of decimal places.

```
SELECT ROUND(123.456, 2) FROM rounded_value;
```

round_value
123.46

CEIL():
Returns the smallest integer greater than or equal to a number (rounds up).

```
SELECT CEIL(6.5) AS ceiling_value;
```

ceiling_value
7

FLOOR():
Returns the smallest integer greater than or equal to a number (rounds up).

```
SELECT FLOOR(6.5) AS floor_value;
```

floor_value
6