



Entity Relationship Diagram

Electronic Product Service System

SEMESTER PROJECT : ERD ,NORMALIZATION and SQL

BS-IT -IV(M)

COURSE CODE:CC-215

DATABASE SYSTEMS

SUBMITTED BY:

EMAN FATIMA-110784

2023-2027

PROJECT GROUP:

GROUP-4

SUBMITTED TO :

MISS SEHRISH KHAN

.....

DEPARTRMENT OF COMPUTER SCIENCES(BS-IT)

PROBLEM STATEMENT

Electronic Product Service System:

This system supports sales of electronic products like laptops and smartphones. Products come with basic or extended warranties. When a customer initiates a service request, the system logs the device, issue, technician assigned, repair history, and whether replacement parts were used. It also flags serial numbers of products that frequently fail for quality control. The system integrates manufacturer recall data to auto-notify customers affected.

Method:

The method for making Entity Relationship Diagram for statement problem is as following:

- Read statement carefully
- Identify Entities
- Identify Attributes
- Identify Relationships
- Check Cardinality
- Make Diagram

Core Entities

Strong entities are:

- Customer
- Product
- Technician
- Manufacturer

Weak entities are:

- Warranty
- ServiceRequest
- RepairHistory
- ReplacementPartUsed

Attributes

1. Product

- ProductID (primary key)
- ProductType (e.g., laptop, smartphone)
- ModelNumber

- SerialNumber
- Description
- ReleaseDate
- Price

2. Customer

- CustomerID (primary key)
- Name
- Email
- Phone
- Address
- PostalCode

3. Technician

- TechnicianID (primary key)
- Name
- Expertise
- ContactInformation

4. Manufacturer

- ManufacturerID (primary key)
- Name
- Address
- ContactInformation

1. Warranty

(weak entity, dependent on Product)

- WarrantyID (partial key)
- ProductID (foreign key, part of composite key)
- WarrantyType (basic or extended)
- CoverageDetails

2. ServiceRequest

(weak entity, dependent on Product and Customer)

- ServiceRequestID (partial key)
- ProductID (foreign key, part of composite key)
- CustomerID (foreign key, part of composite key)
- IssueDescription
- RequestDate

3. Repair History

(weak entity, dependent on ServiceRequest)

- RepairHistoryID (partial key)

- ServiceRequestID (foreign key, part of composite key)
- RepairDate
- Description
- Cost
- TechnicianID (foreign key)

4. ReplacementPartUsed

(weak entity, dependent on RepairHistory)

- ReplacementPartUsedID (partial key)
- RepairHistoryID (foreign key, part of composite key)
- ReplacementPartID (foreign key)
- Quantity

Relationships

1. **Product** has **Warranty** (one-to-one)
2. **Customer** makes **ServiceRequest** (one-to-many)
3. **ServiceRequest** is for **Product** (many-to-one)
4. **ServiceRequest** is assigned to **Technician** (many-to-one)
5. **ServiceRequest** has **RepairHistory** (one-to-many)
6. **RepairHistory** uses **ReplacementPartUsed** (many-to-many)
7. **Product** is manufactured by **Manufacturer** (many-to-one)

Critical Relationship

- **Purchase**(1:N)Customer purchases product
- **Has**(1:1)Mandatory warranty link
- **Generates**(1:N)Service request to repair history
- **Triggers**(M:N)Recall service request association

Cardinality

1. A **product** can have one warranty.
2. A **customer** can make many service requests.

3. A **service request** is for one **product**.
4. A **service request** is assigned to one **technician**.
5. A **service request** can have many **repair histories**.
6. A **repair history** can use many **replacement parts**.

Assumptions

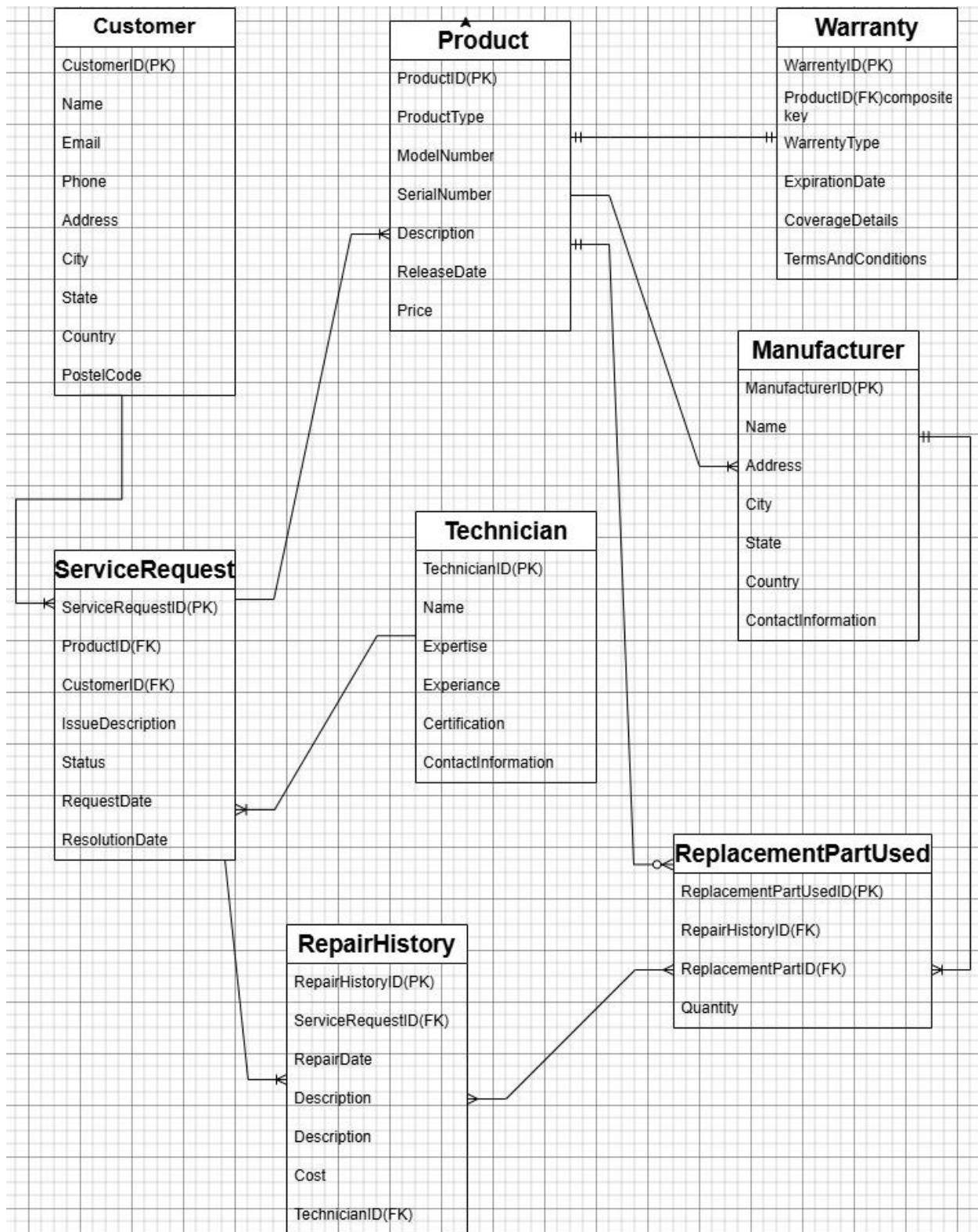
1. Each **product** has a unique serial number.
2. Each **customer** has a unique email address.
3. Each **technician** has a unique name and expertise.
4. Each **replacement part** has a unique part number.

Diagram

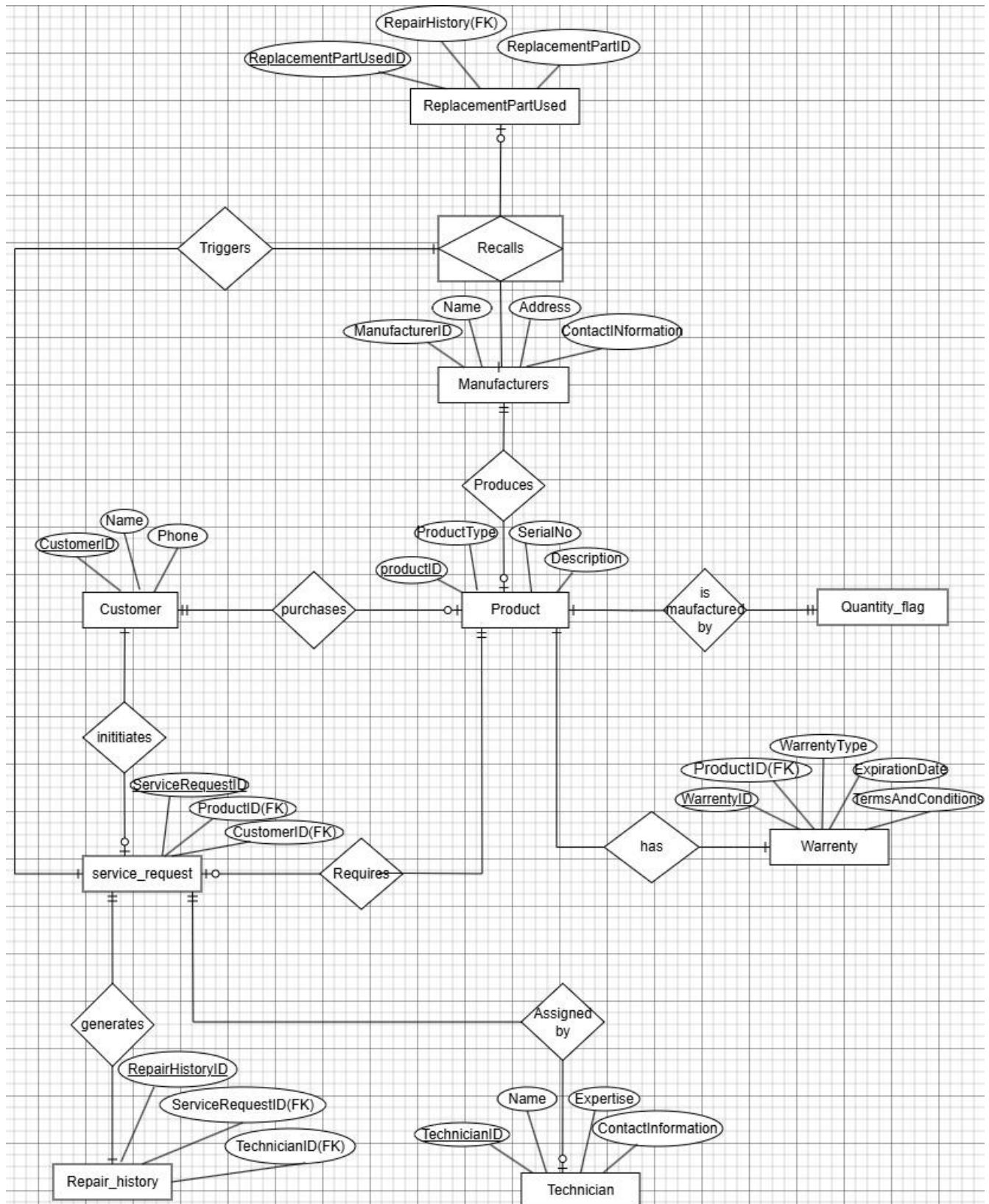


ERD.drawio

Crow's Foot Model



Chen Model



Normalization

	A	B	C	D	E	F	G	H	I	J	K	L	M	
1	CustomerID	CustomerName	ProductID	ProductName	SerialNumber	WarrantyType	ServiceRequestID	Issue	RequestDate	RepairID	RepairDate	RepairDescription	TechnicianName	
2	1	Alice	101	Laptop A	LAP123	Basic	SR001	Screen issue	2025-05-01	RP001	2025-05-03	Screen replaced	Bob	
3	1	Alice	101	Laptop A	LAP123	Basic	SR002	Keyboard fault	2025-05-05	RP002	2025-05-07	Keyboard replaced	Charlie	
4	2	Bob	205	Phone X	PHN456	Extended	SR003	Battery drain	2025-05-08	RP003	2025-05-10	Battery replaced	Bob	
5														
6														
7														

Conceptual Unnormalized Table:

1st Normal Form(1NF):

	A	B	C	D	E	F	G	H	I	J	K	L	M	
1	CustomerID	CustomerName	ProductID	ProductName	SerialNumber	WarrantyType	ServiceRequestID	Issue	RequestDate	RepairID	RepairDate	RepairDescription	TechnicianName	
2	1	Alice	101	Laptop A	LAP123	Basic	SR001	Screen issue	2025-05-01	RP001	2025-05-03	Screen replaced	Bob	
3	1	Alice	101	Laptop A	LAP123	Basic	SR002	Keyboard fault	2025-05-05	RP002	2025-05-07	Keyboard replaced	Charlie	
4	2	Bob	205	Phone X	PHN456	Extended	SR003	Battery drain	2025-05-08	RP003	2025-05-10	Battery replaced	Bob	
5														
6														
7														

2nd Normal Form(2NF):

CustomerProductWarranty Table:

	A	B	C	D	E	F	G
1	CustomerID	ProductID	CustomerName	ProductName	SerialNumber	WarrantyType	
2		1	101	Alice	Laptop A	LAP123	Basic
3		2	205	Bob	Phone X	PHN456	Extended
4							
5							
6							

ServiceRequestRepair Table:

L5	A	B	C	D	E	F	G	H	I	J
1	CustomerID	ProductID	ServiceRequestID	Issue	RequestDate	RepairID	RepairDate	RepairDescription	TechnicianName	
2	1	101	SR001	Screen issue	2025-05-01	RP001	2025-05-03	Screen replaced	Bob	
3	1	101	SR002	Keyboard fault	2025-05-05	RP002	2025-05-07	Keyboard replaced	Charlie	
4	2	205	SR003	Battery drain	2025-05-08	RP003	2025-05-10	Battery replaced	Bob	
5										
6										
7										

3rd Normal Form(3NF):

CustomerProduct Table:

H6	A	B	C
1	CustomerID	ProductID	
2	1	101	
3	2	205	
4			
5			

Customer Details Table:

	A	B	C
1	CustomerID	CustomerName	
2	1	Alice	
3	2	Bob	
4			
5			

Product Details Table:

H12	A	B	C	D	E	F
1	ProductID	ProductName	SerialNumber	ManufacturerID (FK)		
2	101	Laptop A	LAP123	MFR01		
3	205	Phone X	PHN456	MFR02		
4						
5						
6						

Warranty Product Table:

	A	B	C	D
1	WarrantyID	ProductID	WarrantyType	
2	W001	101	Basic	
3	W002	205	Extended	
4				
5				

ServiceRequestIssue Table:

	A	B	C	D	E	F	G
1	ServiceRequestID	CustomerID (FK)	ProductID (FK)	Issue	RequestDate	TechnicianID (FK)	
2	SR001		1	101	Screen issue	2025-05-01	T001
3	SR002		1	101	Keyboard fault	2025-05-05	T002
4	SR003		2	205	Battery drain	2025-05-08	T001
5							
6							

RepairLog Table:

	A	B	C	D	E
1	RepairID	ServiceRequestID (FK)	RepairDate	RepairDescription	
2	RP001	SR001	2025-05-03	Screen replaced	
3	RP002	SR002	2025-05-07	Keyboard replaced	
4	RP003	SR003	2025-05-10	Battery replaced	
5					
6					
7					

TechnicianName Table:

G5 | fx

	A	B	C
1	TechnicianID	TechnicianName	
2	T001	Bob	
3	T002	Charlie	
4			
5			

MySQL Queries

Structured Query Language (SQL)

Common SQL data types

DATATYPE	FORMAT	COMMENTS
Numeric	NUMBER (L, D) OR NUMERIC (L, D)	The declaration NUMBER (7,2) or NUMERIC (7,2) indicates that numbers will be stored with two decimal places and may be up to seven digits long, including the sign and the decimal place (For example, 12.32 or -134.99).
	INTEGER	May be abbreviated as INT. Integers are (whole) counting numbers, so they cannot be used if you want to store numbers that require decimal places.
	SMALL INT	Like INTEGER but limited to integer values up to six digits. If your integer values are relatively small, use SMALLINT instead of INT.
	DECIMAL (L, D)	Like the NUMBER specification, but the storage length is a minimum specification. that is, greater lengths are acceptable, but smaller ones are not. DECIMAL (9,2), DECIMAL (9), and DECIMAL are all acceptable.

Character	CHAR (L)	Fixed-length character data for up to 255 characters. If you store strings that are not as long as the CHAR parameter value, the remaining spaces are left unused. therefore, if you specify CHAR (25), strings such as Smith and Katzenjammer are each stored as 25 characters. However, a U.S. area code is always three digits long, so CHAR (3) would be appropriate if you wanted to store such codes.
	VARCHAR(L) OR SVARCHAR2(L)	Variable-length character data. the designation VARCHAR2(25) or SVARCHAR (25) will let you store characters up to 25 characters long. However, unlike CHAR, VARCHAR will not leave unused spaces. oracle automatically converts VARCHAR to VARCHAR2.
Date	DATE	Stores dates in the Julian date format.

SQL data types and keys

➤ Keys:

- PK (Primary Key):
 - ✚ Uniquely identifies each row in a table.
 - ✚ Cannot be NULL.
 - ✚ Each table typically has one primary key.
- FK (Foreign Key):
 - ✚ A field (or set of fields) that refers to the primary key in another table.
 - ✚ Ensures referential integrity between related tables.

➤ Data Types:

- CHAR(n):
 - ✚ Fixed-length character string.
 - ✚ Always stores exactly n characters (padded with spaces if shorter).
- VARCHAR(n):
 - ✚ Variable-length character string.
 - ✚ Stores up to n characters.
 - ✚ In Oracle, VARCHAR is automatically treated as VARCHAR2.
- NUMBER (p, s): (Oracle)
 - ✚ Numeric type with p total digits and s digits after the decimal point.
 - ✚ Example: NUMBER (9,2) allows 7 digits before the decimal and 2 after.
- NUMERIC (p, s):
 - ✚ Same idea as NUMBER, but used in systems like PostgreSQL or SQL Server.
 - ✚ Ensures precision for exact values.

- INT / INTEGER:
 - Stores whole numbers.
 - In Oracle, it's converted to NUMBER.
 - SMALLINT:
 - Like INT but for smaller range of values (uses less storage).
 - Also converted to NUMBER in Oracle.
 - DATE:
 - Stores date values.
 - Common formats:
 - DD-MON-YYYY (e.g., 02-MAY-2025)
 - MM/DD/YYYY (e.g., 05/02/2025)

USEFUL SQL COMMANDS

- ✓ SHOW DATABASES;
- ✓ USE database name;
- ✓ SHOW TABLES;
- ✓ create table table_name(attribute_id_1 type(domain) primary key, attribute_2 type(domain),...,n);
- ✓ describe table_name;
- ✓ select * from table_name;
- ✓ alter table table_name ADD constraint fk_table_name foreign key (attribute) references ref_table(ref_attribute);
- ✓ insert into table_name value('val.1', 'val.2', 'val.3',...,n);
- ✓ alter table table_name ADD attribute_name type(domain);
- ✓ update table_name set attribute='value' WHERE attribute_PK='target-value';

IMPLEMENTATIONS OF DIFFERENT COMMANDS ON MYSQL

SQL COMMANDS

- CREATE DATABASE
 - DESCRIBE TABLE (DESC)

- INSERT INTO
- ADD VALUES IN TABLES
- ADD FOREIGN KEY OR MULTIPLE FOREIGN KEY
- SELECT COMMAND
- UPDATE
- AS
- DSITINCT
- ORDER BY
- WHERE clause
- SELECT COMMAND
- ARITHMETIC OPERATORS
- RELATIONAL OPERATORS
- LOGICAL OPERATORS
- AND BETWEEN
- IN
- LIKE
- NULL and NOT NULL
- AGREGRATE FUNCTIONS
- GROUP BY
- HAVING
- JOIN
 - INNER
 - OUTER
 - RIGHT
 - LEFT
- SHOW PRIVILEGES
- USER GRANTS
 - CREATE USER
 - CREATE ROLE
 - GRANT ALL PRIVILEGES
 - SHOW GRANTS
- DELETE TABLE / DATABASE

CREATE DATABASE

Procedure to create Database in SQL command Line:

1. Open the SQL Command Line Client.
 2. Enter the password.
 3. Enter the command to show the databases→ **SHOW DATABASES;**
 4. If database is already existing then use command to manipulate in target database.
- **USE database_name;**
 - Database name: market;
 - Example: **use market;**

The screenshot shows a terminal window titled "MySQL 8.0 Command Line Cli". The session starts with the MySQL prompt and welcome message. The user attempts to create a database named "Market" using the command "CREATE DATABASE Market;". This results in an error message: "ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'DATABASE Market' at line 1". After the error, the user runs "SHOW databases;" which lists the available databases: "dept", "information_schema", "market", "mysql", "performance_schema", and "sys". Finally, the user successfully switches to the "market" database using "USE market;" and creates a table named "customer" with the specified schema.

```
MySQL 8.0 Command Line Cli + v
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.42 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE Market;
Query OK, 1 row affected (0.03 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| dept    |
| information_schema |
| market   |
| mysql    |
| performance_schema |
| sys     |
+-----+
6 rows in set (0.00 sec)

mysql> USE market;
Database changed
mysql> CREATE TABLE customer(c_id INT(6) PRIMARY KEY, Name VARCHAR(25), Address VARCHAR(50), Email VARCHAR(30), PostelCode INT);
Query OK, 0 rows affected, 1 warning (0.07 sec)
```

- **Show tables;**
- If you have just created your database so create tables as; **CREATE TABLE table_name();**
- **CREATE TABLE customer(c_id INT(6) PRIMARY KEY, Name VARCHAR(25), Address VARCHAR(50), Email VARCHAR(30), PostelCode INT);**
- Check the description of the table as; **DESC table_name;**
- **DESCRIBE customer;**

```

MySQL 8.0 Command Line Cli + X
Database changed
mysql> CREATE TABLE customer(c_id INT(6) PRIMARY KEY,Name VARCHAR(25),Address VARCHAR(50), Email VARCHAR(30), PostelCode INT);
Query OK, 0 rows affected, 1 warning (0.07 sec)

mysql> DESCRIBE customer;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| c_id   | int    | NO   | PRI | NULL    |       |
| Name   | varchar(25) | YES  |     | NULL    |       |
| Address | varchar(50) | YES  |     | NULL    |       |
| Email  | varchar(30) | YES  |     | NULL    |       |
| PostelCode | int    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.04 sec)

mysql> CREATE TABLE product(
    -> product_id INT(6) PRIMARY KEY,
    -> Model_no VARCHAR(50),
    -> Product_type VARCHAR(50),
    -> Price DECIMAL(10,2),
    -> Serial_no VARCHAR(50)
    -> );
Query OK, 0 rows affected, 1 warning (0.05 sec)

mysql> DESCRIBE product;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| product_id | int    | NO   | PRI | NULL    |       |
| Model_no   | varchar(50) | YES  |     | NULL    |       |
| Product_type | varchar(50) | YES  |     | NULL    |       |
| Price      | decimal(10,2) | YES  |     | NULL    |       |
| Serial_no  | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

- Also create other tables using same command.
- Technician and manufacturer table;

```

MySQL 8.0 Command Line Cli + X
mysql> DESCRIBE technician;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| technician_id | int    | NO   | PRI | NULL    |       |
| Name   | varchar(25) | YES  |     | NULL    |       |
| Expertise | varchar(50) | YES  |     | NULL    |       |
| Contact_info | varchar(70) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> CREATE TABLE manufacturer(manufacturer_id INT PRIMARY KEY, Name
    -> VARCHAR(25), Address Va
    -> VARCHAR(25), Address Va
    -> Contact_info VARCHAR(50));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Va
VARCHAR(25), Address Va
Contact_info VARCHAR(50))' at line 2
mysql> CREATE TABLE manufacturer(manufacturer_id INT PRIMARY KEY, Name VARCHAR(25), Address VARCHAR(50), Contact_info VARCHAR(50));
Query OK, 0 rows affected (0.04 sec)

mysql> DESCRIBE manufacturer;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| manufacturer_id | int    | NO   | PRI | NULL    |       |
| Name   | varchar(25) | YES  |     | NULL    |       |
| Address | varchar(50) | YES  |     | NULL    |       |
| Contact_info | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

```

- Warranty table;

```

MySQL 8.0 Command Line Cli + X
mysql> desc warrenty;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| warrenty_id | int    | NO   | PRI | NULL    |       |
| warrenty_type | varchar(50) | YES  |     | NULL    |       |
| coverage_details | varchar(50) | YES  |     | NULL    |       |
| product_id | int    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

```

- Service request table;
- Alter the table attributes by using the command; **ALTER TABLE servicerequest ADD c_id int(6);**

```

MySQL 8.0 Command Line Cli × + ▾

mysql> CREATE TABLE servicerequest(service_req_id INT PRIMARY KEY,issue_desc VARCHAR(100),Request_date DATE);
Query OK, 0 rows affected (0.05 sec)

mysql> ALTER TABLE servicerequest ADD product_id INT(6), c_id INT(6);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'c_id INT(6)' at line 1
mysql> ALTER TABLE servicerequest ADD product_id INT(6);
Query OK, 0 rows affected, 1 warning (0.08 sec)
Records: 0 Duplicates: 0 Warnings: 1

mysql> ALTER TABLE servicerequest ADD c_id int(6);
Query OK, 0 rows affected, 1 warning (0.07 sec)
Records: 0 Duplicates: 0 Warnings: 1

mysql> DESC servicerequest;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| service_req_id | int | NO | PRI | NULL |
| issue_desc | varchar(100) | YES | | NULL |
| Request_date | date | YES | | NULL |
| product_id | int | YES | | NULL |
| c_id | int | YES | | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

- Repair history table;

```

MySQL 8.0 Command Line Cli × + ▾

mysql> desc repairhistory;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| repair_history_id | int | NO | PRI | NULL |
| Repair_date | date | YES | | NULL |
| Description | varchar(100) | YES | | NULL |
| cost | decimal(10,2) | YES | | NULL |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> ALTER TABLE servicerequest
->
->
-> ALTER TABLE repairhistory ADD service_req_id int(6);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'TABLE repairhistory ADD service_req_id int(6)' at line 5
mysql> ALTER TABLE repairhistory ADD service_req_id int(6);
Query OK, 0 rows affected, 1 warning (0.08 sec)
Records: 0 Duplicates: 0 Warnings: 1

mysql> ALTER TABLE repairhistory ADD technician_id int(6);
Query OK, 0 rows affected, 1 warning (0.08 sec)
Records: 0 Duplicates: 0 Warnings: 1

mysql> desc repairhistory;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| repair_history_id | int | NO | PRI | NULL |
| Repair_date | date | YES | | NULL |
| Description | varchar(100) | YES | | NULL |
| cost | decimal(10,2) | YES | | NULL |
| service_req_id | int | YES | | NULL |
| technician_id | int | YES | | NULL |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

```

- Replacement part used table;

```

MySQL 8.0 Command Line Cli × + ▾

mysql> CREATE TABLE replacementpartused (rpu_id int PRIMARY KEY , repair_history_id int, quantity int);
Query OK, 0 rows affected (0.04 sec)

mysql> desc replacementpartused;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| rpu_id | int | NO | PRI | NULL |
| repair_history_id | int | YES | | NULL |
| quantity | int | YES | | NULL |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

CREATE TABLES

It's used for the creation of tables. Evolving the following statement;

Syntax:

```
CREATE TABLE table_name (column_1,column_2,.....);
```

- **CREATE TABLE customer(c_id INT(6) PRIMARY KEY,Name VARCHAR(25),Address VARCHAR(50), Email VARCHAR(30), PostelCode INT);**
- Check the description of the table as; **DESC table_name;**
- **DESCRIBE customer;**

The screenshot shows a terminal window for MySQL 8.0 Command Line Client. It displays the creation of two tables: 'customer' and 'product'. After each table is created, its description is printed using the 'DESCRIBE' command.

```
MySQL 8.0 Command Line Cli X + - □ ×
Database changed
mysql> CREATE TABLE customer(c_id INT(6) PRIMARY KEY,Name VARCHAR(25),Address VARCHAR(50), Email VARCHAR(30), PostelCode INT);
Query OK, 0 rows affected, 1 warning (0.07 sec)

mysql> DESCRIBE customer;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| c_id | int | NO | PRI | NULL |          |
| Name | varchar(25) | YES |     | NULL |          |
| Address | varchar(50) | YES |     | NULL |          |
| Email | varchar(30) | YES |     | NULL |          |
| PostelCode | int | YES |     | NULL |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.04 sec)

mysql> CREATE TABLE product(
    > product_id INT(6) PRIMARY KEY,
    > Model_no VARCHAR(50),
    > Product_type VARCHAR(50),
    > Price DECIMAL(10,2),
    > Serial_no VARCHAR(50)
    > );
Query OK, 0 rows affected, 1 warning (0.05 sec)

mysql> DESCRIBE product;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| product_id | int | NO | PRI | NULL |          |
| Model_no | varchar(50) | YES |     | NULL |          |
| Product_type | varchar(50) | YES |     | NULL |          |
| Price | decimal(10,2) | YES |     | NULL |          |
| Serial_no | varchar(50) | YES |     | NULL |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Use of DESCRIBE COMMAND

It's also can be used as the DESC shortly.

Syntax:

```
DESC table_name;
```

```

mysql> desc customer;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| c_id | int | NO | PRI | NULL |          |
| Name | varchar(25) | YES |     | NULL |          |
| Address | varchar(50) | YES |     | NULL |          |
| Email | varchar(30) | YES |     | NULL |          |
| PostelCode | int | YES |     | NULL |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)

```

INSERT COMMAND

Syntax;

INSERT INTO table_name VALUES (column_1 , values_C.....);

- Insert values in tables by using command as;

```

INSERT INTO manufacturer values
('101','Ali','skp','0321-1234567'),
('102','Alyan','lhr','0321-1234569'),
('103','Arman','fsd','0321-1234568');

```

```

4 rows in set (0.00 sec)

mysql> INSERT INTO manufacturer values('101','Ali','skp','0321-1234567'),('101','Alyan','lhr','0321-1234569'),('103','Arman','fsd','0321-1234568');
ERROR 1062 (23000): Duplicate entry '101' for key 'manufacturer.PRIMARY'
mysql> INSERT INTO manufacturer values('101','Ali','skp','0321-1234567'),('102','Alyan','lhr','0321-1234569'),('103','Arman','fsd','0321-1234568');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> DESC manufacturer;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| manufacturer_id | int | NO | PRI | NULL |          |
| Name | varchar(25) | YES |     | NULL |          |
| Address | varchar(50) | YES |     | NULL |          |
| Contact_info | varchar(50) | YES |     | NULL |          |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select *from manufacturer;
+-----+-----+-----+-----+
| manufacturer_id | Name | Address | Contact_info |
+-----+-----+-----+-----+
| 101 | Ali | skp | 0321-1234567 |
| 102 | Alyan | lhr | 0321-1234569 |
| 103 | Arman | fsd | 0321-1234568 |
+-----+-----+-----+-----+
3 rows in set (0.02 sec)

mysql> desc customer;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| c_id | int | NO | PRI | NULL |          |
| Name | varchar(25) | YES |     | NULL |          |
| Address | varchar(50) | YES |     | NULL |          |
| Email | varchar(30) | YES |     | NULL |          |
| PostelCode | int | YES |     | NULL |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> INSERT INTO customer VALUES('1','Azka','fqd','azka09@gmail.com','345'),('2','zayan','skp','zayan08@gmail.com','567'),('3','Anaiza','lhr','anaiza777@gmail.com','897');
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select *from customer;
+-----+-----+-----+-----+-----+
| c_id | Name | Address | Email | PostelCode |
+-----+-----+-----+-----+-----+
| 1 | Azka | fqd | azka09@gmail.com | 345 |
| 2 | zayan | skp | zayan08@gmail.com | 567 |
| 3 | Anaiza | lhr | anaiza777@gmail.com | 897 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc servicerequest;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+

```

ADD Values in tables

- Insert values in tables by using command as;

```

INSERT INTO manufacturer values
('101','Ali','skp','0321-1234567'),
('102','Alyan','lhr','0321-1234569'),
('103','Arman','fsd','0321-1234568');

```

```

MySQL 8.0 Command Line Cli + X
4 rows in set (0.00 sec)

mysql> INSERT INTO manufacturer values('101','Ali','skp','0321-1234567'),('101','Alyan','lhr','0321-1234569'),('103','Arman','fsd','0321-1234568');
ERROR 1062 (23000): Duplicate entry '101' for key 'manufacturer.PRIMARY'
mysql> INSERT INTO manufacturer values('101','Ali','skp','0321-1234567'),('102','Alyan','lhr','0321-1234569'),('103','Arman','fsd','0321-1234568');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
mysql> DESC manufacturer;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| manufacturer_id | int | NO | PRI | NULL |          |
| Name | varchar(25) | YES |     | NULL |          |
| Address | varchar(50) | YES |     | NULL |          |
| Contact_info | varchar(50) | YES |     | NULL |          |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select *from manufacturer;
+-----+-----+-----+-----+
| manufacturer_id | Name | Address | Contact_info |
+-----+-----+-----+-----+
| 101 | Ali | skp | 0321-1234567 |
| 102 | Alyan | lhr | 0321-1234569 |
| 103 | Arman | fsd | 0321-1234568 |
+-----+-----+-----+-----+
3 rows in set (0.02 sec)

mysql> desc customer;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| c_id | int | NO | PRI | NULL |          |
| Name | varchar(25) | YES |     | NULL |          |
| Address | varchar(50) | YES |     | NULL |          |
| Email | varchar(30) | YES |     | NULL |          |
| PostelCode | int | YES |     | NULL |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> INSERT INTO customer VALUES('1','Azka','fqd','azka09@gmail.com','345'), ('2','zayan','skp','zayan08@gmail.com','567'), ('3','Anaiza','lhr','anaiza777@gmail.com','697');
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0
mysql> select *from customer;
+-----+-----+-----+-----+-----+
| c_id | Name | Address | Email | PostelCode |
+-----+-----+-----+-----+-----+
| 1 | Azka | fqd | azka09@gmail.com | 345 |
| 2 | zayan | skp | zayan08@gmail.com | 567 |
| 3 | Anaiza | lhr | anaiza777@gmail.com | 897 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc servicerequest;
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+

```

Same as;

```

MySQL 8.0 Command Line Cli + X
3 rows in set (0.00 sec)

mysql> desc servicerequest;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| service_req_id | int | NO | PRI | NULL |          |
| issue_desc | varchar(100) | YES |     | NULL |          |
| Request_date | date | YES |     | NULL |          |
| product_id | int | YES | MUL | NULL |          |
| c_id | int | YES | MUL | NULL |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> INSERT INTO servicerequest VALUES('401','Resolved','2024-11-12','11','screen not turning on'), ('402','In progress','2024-01-18','12','motor making noise'), ('403','May take time','2024-03-14','13','technical faults');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails ('maindb`.`servicerequest' FOREIGN KEY ('product_id') REFERENCES 'product' ('product_id'))
mysql> INSERT INTO servicerequest VALUES('401','Resolved','2024-11-12','11','1'), ('402','In progress','2024-01-18','12','2'), ('403','May take time','2024-03-14','13','3');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails ('maindb`.`servicerequest' FOREIGN KEY ('product_id') REFERENCES 'product' ('product_id'))
mysql> desc product;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| product_id | int | NO | PRI | NULL |          |
| Model_no | varchar(50) | YES |     | NULL |          |
| Product_type | varchar(50) | YES |     | NULL |          |
| Price | decimal(10, 2) | YES |     | NULL |          |
| Serial_no | varchar(50) | YES |     | NULL |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> INSERT INTO product VALUES ('301', 'MOD-A1', 'Smartphone', '899.99', 'SN-XYZ001'), ('302', 'MOD-B2', 'Laptop', '1499.50', 'SN-ABC202'), ('303', 'MOD-C3', 'Headphones', '199.00', 'SN-QWE345');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
mysql> select *from product;
+-----+-----+-----+-----+-----+
| product_id | Model_no | Product_type | Price | Serial_no |
+-----+-----+-----+-----+-----+
| 301 | MOD-A1 | Smartphone | 899.99 | SN-XYZ001 |
| 302 | MOD-B2 | Laptop | 1499.50 | SN-ABC202 |
| 303 | MOD-C3 | Headphones | 199.00 | SN-QWE345 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc servicerequest;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| service_req_id | int | NO | PRI | NULL |          |
| issue_desc | varchar(100) | YES |     | NULL |          |
| Request_date | date | YES |     | NULL |          |
| product_id | int | YES | MUL | NULL |          |
| c_id | int | YES | MUL | NULL |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> INSERT INTO servicerequest VALUES('401','Resolved','2024-11-12','301','1'), ('402','In progress','2024-01-18','302','2'), ('403','May take time','2024-03-14','303','3');

```

Also same for others;

```

MySQL 8.0 Command Line Cli + X
5 rows in set (0.00 sec)

mysql> INSERT INTO servicerequest VALUES('401','Resolved','2024-11-12','301','1'),('402','In progress','2024-01-18','302','2'),('403','May take time','2024-03-14','303','3');
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select *from servicerequest;
+-----+-----+-----+-----+
| service_req_id | issue_desc | Request_date | product_id | c_id |
+-----+-----+-----+-----+
| 401 | Resolved | 2024-11-12 | 301 | 1 |
| 402 | In progress | 2024-01-18 | 302 | 2 |
| 403 | May take time | 2024-03-14 | 303 | 3 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc replacementpartused;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| rpu_id | int | NO | PRI | NULL |
| repair_history_id | int | YES | MUL | NULL |
| quantity | int | YES | MUL | NULL |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc repairhistory;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| repair_history_id | int | NO | PRI | NULL |
| Repair_date | date | YES | MUL | NULL |
| Description | varchar(100) | YES | MUL | NULL |
| cost | decimal(10,2) | YES | MUL | NULL |
| service_req_id | int | YES | MUL | NULL |
| technician_id | int | YES | MUL | NULL |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> desc technician;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| technician_id | int | NO | PRI | NULL |
| Name | varchar(25) | YES | MUL | NULL |
| Expertise | varchar(25) | YES | MUL | NULL |
| Contact_info | varchar(70) | YES | MUL | NULL |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> INSERT INTO technician VALUES ('604', 'Liam O'Connor', 'Aerospace Avionics Repair', 'liam.avionics@skytech.ie | +353 87 123 4567'), ('605', 'Sofia Rossi', 'Vintage Watch Restoration', 'sofia.restoration@horology.it (Turin Works shop)'), ('606', 'Amir Al-Farsi', 'Oil Rig Industrial Sensors', '+966 55 678 9012 | Emergency Line: 1999');
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select *from technician;
+-----+-----+-----+
| technician_id | Name | Expertise | Contact_info |
+-----+-----+-----+

```

Or

```

MySQL 8.0 Command Line Cli + X
5 rows in set (0.00 sec)

mysql> select *from technician;
+-----+-----+-----+-----+
| technician_id | Name | Expertise | Contact_info |
+-----+-----+-----+-----+
| 604 | Liam O'Connor | Aerospace Avionics Repair | liam.avionics@skytech.ie | +353 87 123 4567 |
| 605 | Sofia Rossi | Vintage Watch Restoration | sofia.restoration@horology.it (Turin Workshop) |
| 606 | Amir Al-Farsi | Oil Rig Industrial Sensors | +966 55 678 9012 | Emergency Line: 1999 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> INSERT INTO technicians
-> VALUES
-> ('404', 'Liam O'Connor', 'Aerospace Avionics Repair', 'liam.avionics@skytech.ie | +353 87 123 4567'),
-> ('605', 'Sofia Rossi', 'Vintage Watch Restoration', 'sofia.restoration@horology.it (Turin Workshop)'),
-> ('606', 'Amir Al-Farsi', 'Oil Rig Industrial Sensors', '+966 55 678 9012 | Emergency Line: 1999');
ERROR 1146 (42000): Table "market.technicians" doesn't exist
mysql> INSERT INTO repairhistory VALUES ('1001', '2025-05-15', 'Screen replacement for device X203', '89.99', '401', '604'), ('1002', '2025-05-10', 'Battery calibration and optimization', '45.50', '402', '605'), ('1003', '2025-04-28', 'Motherboard diagnostics and component replacement', '299.00', '401', '606');
Query OK, 3 rows affected (0.03 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select *from repairhistory;
+-----+-----+-----+-----+-----+
| repair_history_id | Repair_date | Description | cost | service_req_id | technician_id |
+-----+-----+-----+-----+-----+
| 1001 | 2025-05-15 | Screen replacement for device X203 | 89.99 | 401 | 604 |
| 1002 | 2025-05-10 | Battery calibration and optimization | 45.50 | 402 | 605 |
| 1003 | 2025-04-28 | Motherboard diagnostics and component replacement | 299.00 | 401 | 606 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc replacementpartused;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| rpu_id | int | NO | PRI | NULL |
| repair_history_id | int | YES | MUL | NULL |
| quantity | int | YES | MUL | NULL |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> INSERT INTO replacementpartused VALUES ('11','1001','10'),('12','1002','20'),('13','1003','30');
Query OK, 3 rows affected (0.02 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select *from replacementpartused;
+-----+-----+-----+
| rpu_id | repair_history_id | quantity |
+-----+-----+-----+
| 11 | 1001 | 10 |
| 12 | 1002 | 20 |
| 13 | 1003 | 30 |
+-----+-----+-----+

```

ADD FOREIGN KEY

First of all we add the attribute in the child table as an alias and then reference to the parent PK.

We want to add the FK in the student table from course table here student table is child and course table is parent table. First of all we add the attribute as child in the student table and then reference it.

- Use the following command to add new attribute to the table.

Command: **ALTER TABLE servicerequest ADD c_id int(6);**

- Check the service request table ;

```
MySQL 8.0 Command Line Cli × + ×
Records: 0 Duplicates: 0 Warnings: 1
mysql> DESC servicerequest;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| service_req_id | int | NO | PRI | NULL |       |
| issue_desc | varchar(100) | YES |       | NULL |       |
| Request_date | date | YES |       | NULL |       |
| product_id | int | YES |       | NULL |       |
| c_id | int | YES |       | NULL |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

- Now make it as FK using following commands.
- Command :


```
ALTER TABLE servicerequest ADD CONSTRAINT
fk_servicerequest FOREIGN KEY (product_id) REFERENCES
product(product_id);
```

Or

```
ALTER TABLE servicerequest ADD CONSTRAINT
fk_servicerequest FOREIGN KEY (c_id) REFERENCES
customer(c_id);
```

```
mysql> ALTER TABLE servicerequest ADD CONSTRAINT fk_servicerequest FOREIGN KEY (product_id) REFERENCES product(product_id);
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE servicerequest ADD CONSTRAINT fk_servicerequest FOREIGN KEY (c_id) REFERENCES customer(c_id);
ERROR 1061 (42000): Duplicate key name 'fk_servicerequest'
mysql> desc servicerequest;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| service_req_id | int | NO | PRI | NULL |       |
| issue_desc | varchar(100) | YES |       | NULL |       |
| Request_date | date | YES |       | NULL |       |
| product_id | int | YES | MUL | NULL |       |
| c_id | int | YES |       | NULL |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

- Now for two foreign keys in same table use the command as;

```
ALTER TABLE servicerequest
-> ADD CONSTRAINT fk_servicerequest_customer2
```

```

-> FOREIGN KEY (c_id)
-> REFERENCES customer(c_id)
-> ON DELETE CASCADE;

```

```

mysql> ALTER TABLE servicerequest
-> ADD CONSTRAINT fk_servicerequest_customer2 -- Unique constraint name
-> FOREIGN KEY (c_id)
-> REFERENCES customer(c_id)
-> ON DELETE CASCADE; -- Optional: Add referential action
Query OK, 0 rows affected (0.11 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc servicerequest;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| service_req_id | int | NO | PRI | NULL |       |
| issue_desc | varchar(100) | YES |       | NULL |       |
| Request_date | date | YES |       | NULL |       |
| product_id | int | YES | MUL | NULL |       |
| c_id | int | YES | MUL | NULL |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

SELECT * command

This command is used for the displaying all and every single attribute's values on the command prompt.

Syntax;

```
SELECT *FROM table_name;
```

```

MySQL> select *from customer;
+----+-----+-----+-----+-----+
| c_id | Name | Address | Email | PostelCode |
+----+-----+-----+-----+-----+
| 1 | Azka | fqd | azka09@gmail.com | 345 |
| 2 | zayan | skp | zayan08@gmail.com | 567 |
| 3 | Anaiza | lhr | anaiza777@gmail.com | 897 |
+----+-----+-----+-----+-----+
3 rows in set (0.02 sec)

MySQL> select *from manufacturer;
+-----+-----+-----+-----+
| manufacturer_id | Name | Address | Contact_info |
+-----+-----+-----+-----+
| 101 | Ali | skp | 0321-1234567 |
| 102 | Alyan | lhr | 0321-1234569 |
| 103 | Arman | fsd | 0321-1234568 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

MySQL> select *from product;
+-----+-----+-----+-----+
| product_id | Model_no | Product_type | Price | Serial_no |
+-----+-----+-----+-----+
| 301 | MOD-A1 | Smartphone | 899.99 | SN-XYZ091 |
| 302 | MOD-B2 | Laptop | 1499.50 | SN-ABC292 |
| 303 | MOD-C3 | Headphones | 199.00 | SN-QWE345 |
+-----+-----+-----+-----+
3 rows in set (0.02 sec)

```

SELECT one/multiple column

This command is beneficial at that support while you are dealing with to come out the single one and the multiple attributes from a table rather than all of the attributes. Only a certain or specific value will be come out by this.

Syntax;

```
SELECT column_name FROM table_name;
```



A screenshot of the MySQL 8.0 Command Line Client window. The query `select rpu_id from replacementpartused;` is run, resulting in a table output:

rpu_id
11
12
13

3 rows in set (0.00 sec)

ALTER command

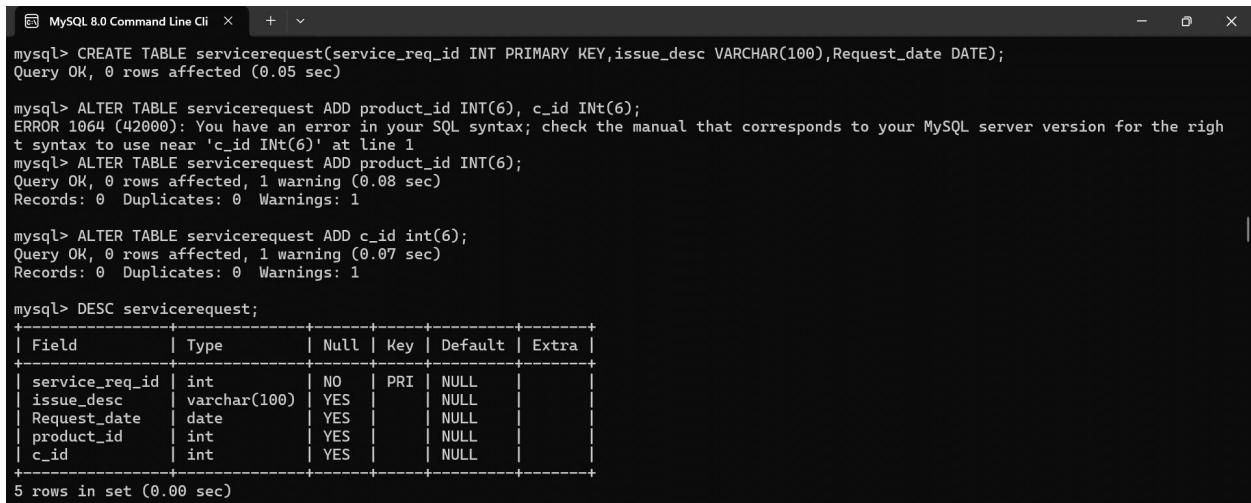
Used to alter the table and including other attributes or inserts more data into the table.

Syntax:

```
ALTER TABLE table_name ADD column_name CONSTRAINT;
```

- Alter the table attributes by using the command;

```
ALTER TABLE servicerequest ADD c_id int(6);
```



A screenshot of the MySQL 8.0 Command Line Client window. It shows the creation of a table and attempts to add a new column:

```
mysql> CREATE TABLE servicerequest(service_req_id INT PRIMARY KEY,issue_desc VARCHAR(100),Request_date DATE);
Query OK, 0 rows affected (0.05 sec)

mysql> ALTER TABLE servicerequest ADD product_id INT(6), c_id INT(6);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'c_id INT(6)' at line 1
mysql> ALTER TABLE servicerequest ADD product_id INT(6);
Query OK, 0 rows affected, 1 warning (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> ALTER TABLE servicerequest ADD c_id int(6);
Query OK, 0 rows affected, 1 warning (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> DESC servicerequest;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| service_req_id | int        | NO   | PRI | NULL    |       |
| issue_desc   | varchar(100) | YES  |     | NULL    |       |
| Request_date | date       | YES  |     | NULL    |       |
| product_id   | int        | YES  |     | NULL    |       |
| c_id         | int        | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Suppose we want to include the already present table EMPLOYEE the PRIMARY KEY. Then for the sake of that purpose we use:

AS COMMAND on MySQL

This is basically used to make an alias of the attribute. It's helpful where you want to access the already attribute name with the new one as you want.

Syntax:

```
SELECT column_name AS ALIAS FROM table_name;
```

A screenshot of the MySQL 8.0 Command Line Client window. The command entered is 'SELECT coverage_details AS information from warrenty;'. The output shows three rows of data under the 'information' alias, which lists warranty details: 'Parts and labor for 1 year', 'Full coverage for 3 years', and 'Accidental damage + 24/7 support'. The total number of rows is 3, displayed in 0.00 seconds.

information
Parts and labor for 1 year
Full coverage for 3 years
Accidental damage + 24/7 support

3 rows in set (0.00 sec)

DISTINCT command

This command is used for duplicity removal from your table if exist.

Syntax:

```
SELECT DISTINCT column_name FROM table_name;
```

A screenshot of the MySQL 8.0 Command Line Client window. The command entered is 'SELECT DISTINCT Expertise from technician;'. The output shows four distinct rows under the 'Expertise' alias, listing different technical expertise areas: 'Aerospace Avionics Repair', 'Vintage Watch Restoration', and 'Oil Rig Industrial Sensors'. The total number of rows is 3, displayed in 0.02 seconds.

Expertise
Aerospace Avionics Repair
Vintage Watch Restoration
Oil Rig Industrial Sensors

3 rows in set (0.02 sec)

WHERE clause in SQL

Where command is work as such like from the condition .

Syntax:

```
SELECT *FROM table_name WHERE column='VALUES';
```

A screenshot of the MySQL 8.0 Command Line Client window. The command entered is 'SELECT *from customer WHERE c_id='2';'. The output shows one row of data from the 'customer' table, where the primary key 'c_id' is 2, and the name is 'zayan'. The total number of rows is 1, displayed in 0.02 seconds.

c_id	Name	Address	Email	PostelCode
2	zayan	skp	zayan08@gmail.com	567

1 row in set (0.02 sec)

ORDER BY command

It's used for the ascending or descending order sorting. By default ascending sort is done by the compiler.

Syntax:

```
SELECT column_name FROM table_name ORDER BY column_2 desc;
```

```
MySQL> select Name ,Address FROM manufacturer;
+-----+-----+
| Name | Address |
+-----+-----+
| Ali  | skp   |
| Alyan | lhr  |
| Arman | fsd  |
+-----+-----+
3 rows in set (0.00 sec)
```

OPERATORS in SQL

Arithmetic operators including (+,-,*,/).

Syntax:

```
SELECT column_name OPERATOR FROM table_name WHERE  
column_name='VALUES' ;
```

Or

```
SELECT column_name FROM table_name OPERATOR any_operation;
```

```
MySQL> select cost*12 FROM repairhistory;
+-----+
| cost*12 |
+-----+
| 1079.88 |
| 546.00  |
| 3588.00 |
+-----+
3 rows in set (0.02 sec)

MySQL> select cost+100 AS Expenditure FROM repairhistory;
+-----+
| Expenditure |
+-----+
| 189.99    |
| 145.50    |
| 399.00    |
+-----+
3 rows in set (0.00 sec)

MySQL> select cost-50 FROM repairhistory;
+-----+
| cost-50 |
+-----+
| 39.99   |
| 4.50    |
| 249.00  |
+-----+
3 rows in set (0.02 sec)
```

Or

```
MySQL> select cost+100 FROM repairhistory WHERE repair_history_id='1002';
+-----+
| cost+100 |
+-----+
| 145.50  |
+-----+
1 row in set (0.00 sec)

MySQL> select cost+100*12 AS Total_Cost FROM repairhistory;
+-----+
| Total_Cost |
+-----+
| 1289.99  |
| 1245.50  |
| 1499.00  |
+-----+
3 rows in set (0.00 sec)
```

RELATIONAL OPERATORS

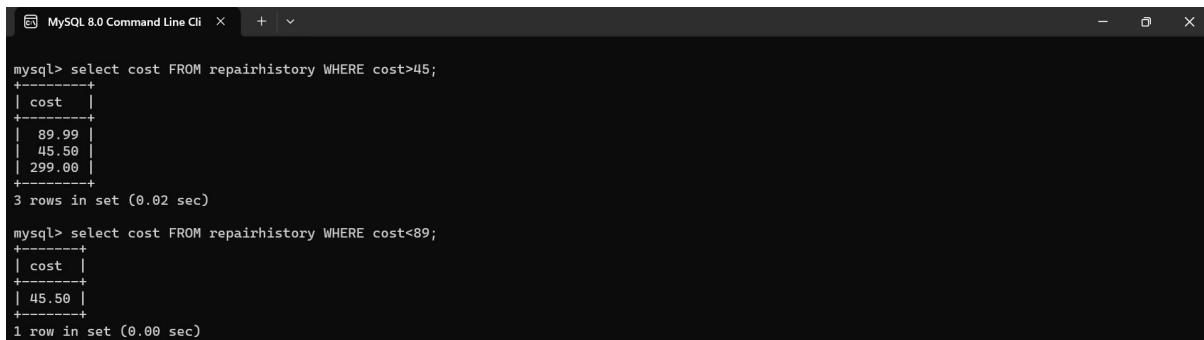
These operators including ($>$, $<$, \geq , \leq , $=$, $!=$).

Syntax:

```
SELECT column_name OPERATOR FROM table_name WHERE  
column_name='VALUES';
```

Or

```
SELECT column_name FROM table_name WHERE column OPERATOR;
```



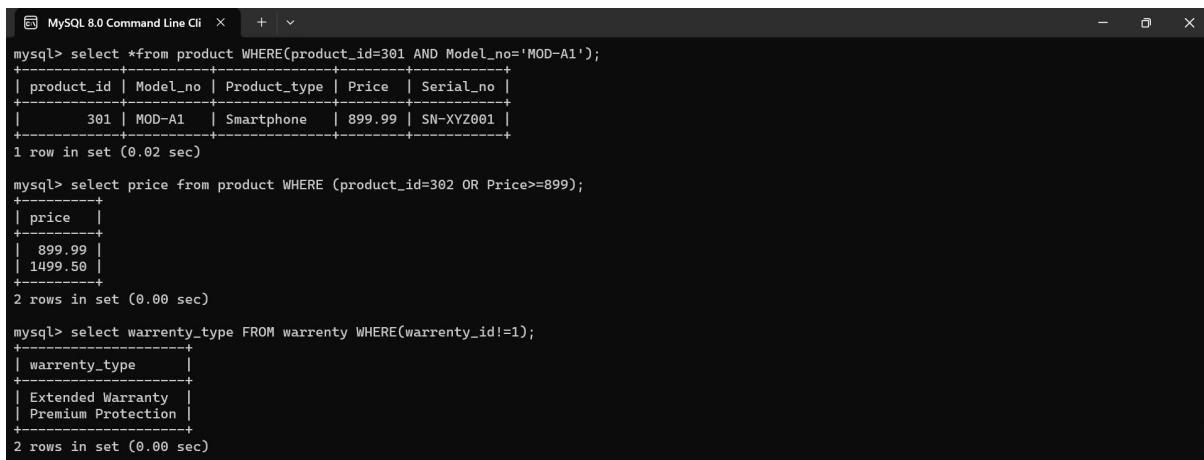
The screenshot shows a MySQL command-line interface window. It displays two SQL queries and their results. The first query is 'select cost FROM repairhistory WHERE cost>45;' and the second is 'select cost FROM repairhistory WHERE cost<89;'. Both queries return the 'cost' column from the 'repairhistory' table.

```
mysql> select cost FROM repairhistory WHERE cost>45;  
+-----+  
| cost |  
+-----+  
| 89.99 |  
| 45.50 |  
| 299.00 |  
+-----+  
3 rows in set (0.02 sec)  
  
mysql> select cost FROM repairhistory WHERE cost<89;  
+-----+  
| cost |  
+-----+  
| 45.50 |  
+-----+  
1 row in set (0.00 sec)
```

LOGICAL OPERATORS

AND ,OR ,NOT.

SQL QUERIES BY AND,OR OPERATOR;



The screenshot shows a MySQL command-line interface window. It displays three SQL queries. The first query is 'select *from product WHERE(product_id=301 AND Model_no='MOD-A1');'. The second query is 'select price from product WHERE (product_id=302 OR Price>=899);'. The third query is 'select warrenty_type FROM warrenty WHERE(warrenty_id!=1);'. The results show the selected columns for each query.

```
mysql> select *from product WHERE(product_id=301 AND Model_no='MOD-A1');  
+-----+-----+-----+-----+  
| product_id | Model_no | Product_type | Price | Serial_no |  
+-----+-----+-----+-----+  
| 301 | MOD-A1 | Smartphone | 899.99 | SN-XYZ001 |  
+-----+-----+-----+-----+  
1 row in set (0.02 sec)  
  
mysql> select price from product WHERE (product_id=302 OR Price>=899);  
+-----+  
| price |  
+-----+  
| 899.99 |  
| 1499.50 |  
+-----+  
2 rows in set (0.00 sec)  
  
mysql> select warrenty_type FROM warrenty WHERE(warrenty_id!=1);  
+-----+  
| warrenty_type |  
+-----+  
| Extended Warranty |  
| Premium Protection |  
+-----+  
2 rows in set (0.00 sec)
```

Use of MULTIPLE AND OPERATOR

```

mysql> select *from manufacturer WHERE (manufacturer_id='101' && Address='skp');
+-----+-----+-----+-----+
| manufacturer_id | Name | Address | Contact_info |
+-----+-----+-----+-----+
| 101 | Ali | skp | 0321-1234567 |
+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql> select *from customer WHERE (c_id='2' OR Address='skp');
+-----+-----+-----+-----+
| c_id | Name | Address | Email | PostelCode |
+-----+-----+-----+-----+
| 2 | zayan | skp | zayan08@gmail.com | 567 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select Model_no FROM product WHERE (product_id='303' AND Price>='199' AND Price<='1499' AND Price BETWEEN 199 AND 1499);
+-----+
| Model_no |
+-----+
| MOD-C3 |
+-----+
1 row in set (0.02 sec)

mysql> select repair_history_id FROM repairhistory WHERE (repair_history_id='1003' OR cost BETWEEN 89 AND 299);
+-----+
| repair_history_id |
+-----+
| 1001 |
| 1003 |
+-----+
2 rows in set (0.00 sec)

```

AND BETWEEN

It's used where you want to evaluate the values from the range. Within that specific range. SQL COMMANDS.

Syntax:

```
SELECT *FROM table_name WHERE(condition_1 AND condition_2  
BETWEEN condition_3);
```

As you can use any column_name rather than the asterisk inside the query.

```

mysql> select Model_no FROM product WHERE (product_id='303' AND Price>='199' AND Price<='1499' AND Price BETWEEN 199 AND 1499);
+-----+
| Model_no |
+-----+
| MOD-C3 |
+-----+
1 row in set (0.02 sec)

mysql> select repair_history_id FROM repairhistory WHERE (repair_history_id='1003' OR cost BETWEEN 89 AND 299);
+-----+
| repair_history_id |
+-----+
| 1001 |
| 1003 |
+-----+
2 rows in set (0.00 sec)

```

IN command

It's uses to come out the certain value from the list of the table. It's basically used to reduce length which is taken by multiple AND or OR operators.

Syntax:

```
SELECT column_name FROM table_name WHERE column_name  
IN(val_1 ,val_2);
```

```
MySQL 8.0 Command Line Cli + - X
mysql> select Repair_date from repairhistory WHERE cost IN(45,299);
+-----+
| Repair_date |
+-----+
| 2025-04-28 |
+-----+
1 row in set (0.00 sec)
```

LIKE command

As if you want to evaluate a specific personality from a table whose spelling you are specifying in the query or any other name whose end or start or at any middle element you'll be mentioned by you. This command is beneficial at that spot while you want to take a name whose character you'll not be known is advanced.

Wildcard Characters:

- %(Represents the single ,null or multiple characters)
- _(Represents a single character)

Syntax:

```
SELECT column_1,column_2 FROM table_name WHERE
column_name LIKE pattern;
```

```
MySQL 8.0 Command Line Cli + - X
mysql> select Name FROM customer WHERE Name LIKE 'A%';
+-----+
| Name |
+-----+
| Azka |
| Anaiza |
+-----+
2 rows in set (0.00 sec)

mysql> select Name FROM customer WHERE Name LIKE '_A%';
+-----+
| Name |
+-----+
| zayan |
+-----+
1 row in set (0.00 sec)
```

NULL

```
MySQL 8.0 Command Line Cli + - X
mysql> select Name from customer WHERE PostelCode IS NULL;
Empty set (0.00 sec)
```

As there is no column contains NULL VALUES so it returns ‘empty set’ ,but command is correct.

AGGREGATION FUNCTIONS:

- COUNT(COLUMN_NAME)
- COUNT(*)
- AVG(COLUMN_NAME)
- MAX(COLUMN_NAME)
- MIN(COLUMN_NAME)
- SUM(COUMN_NAME)

COUNT command

```

mysql> select COUNT(*) FROM customer;
+-----+
| COUNT(*) |
+-----+
|      3   |
+-----+
1 row in set (0.00 sec)

mysql> select COUNT(Price) FROM product;
+-----+
| COUNT(Price) |
+-----+
|      3   |
+-----+
1 row in set (0.02 sec)

```

- **COUNT(*)** will display all the columns which are included in your tables including the null values and also the duplicate values.
- **COUNT(column_name)** will display ONLY the single values and not null values.

AGGREGATE FUNCTIONS

Syntax:

```
SELECT AGGREGATE_FUNCTION(column_name) FROM table_name;
```

AVERAGE

```

mysql> select AVG(cost) FROM repairhistory;
+-----+
| AVG(cost) |
+-----+
| 144.830000 |
+-----+
1 row in set (0.00 sec)

```

MIN , MAX , SUM

```

mysql> select MAX(cost) FROM repairhistory;
+-----+
| MAX(cost) |
+-----+
| 299.00 |
+-----+
1 row in set (0.02 sec)

mysql> select MIN(cost) FROM repairhistory;
+-----+
| MIN(cost) |
+-----+
| 45.50 |
+-----+
1 row in set (0.02 sec)

mysql> select SUM(Price) FROM product;
+-----+
| SUM(Price) |
+-----+
| 2598.49 |
+-----+
1 row in set (0.00 sec)

mysql> select MAX(cost) FROM repairhistory GROUP BY Repair_date ,Description;
+-----+
| MAX(cost) |
+-----+
| 89.99 |
| 45.50 |
| 299.00 |
+-----+
3 rows in set (0.00 sec)

```

GROUP BY

```
MySQL> select MAX(cost) FROM repairhistory GROUP BY Repair_date ,Description;
+-----+
| MAX(cost) |
+-----+
| 89.99 |
| 45.50 |
| 299.00 |
+-----+
3 rows in set (0.00 sec)
```

HAVING command

It's uses as such to apply the conditions indirectly which is not done with the help of where command as while you are dealing with the GROUP BY in sql. So for the sake of that purpose you have to use the HAVING to filter out of data or attributes as you want from the table.

For example:

```
MySQL 8.0 Command Line Cli + x - _ X

mysql> select MAX(cost) FROM repairhistory GROUP BY Repair_date HAVING COUNT(repair_history_id)>2;
+-----+
| MAX(cost) |
+-----+
| 89.99 |
| 45.50 |
| 299.00 |
+-----+
3 rows in set (0.02 sec)

mysql> select MAX(cost) FROM repairhistory GROUP BY Repair_date HAVING COUNT(repair_history_id)>3;
Empty set (0.00 sec)

mysql> select MAX(cost) FROM repairhistory GROUP BY Repair_date HAVING COUNT(repair_history_id)>2;
Empty set (0.00 sec)

mysql> select MAX(cost) FROM repairhistory GROUP BY Repair_date HAVING COUNT(repair_history_id)>1;
Empty set (0.00 sec)

mysql> select MAX(cost) FROM repairhistory GROUP BY Repair_date HAVING COUNT(repair_history_id)>1;
+-----+
| MAX(cost) |
+-----+
| 89.99 |
| 45.50 |
| 299.00 |
+-----+
3 rows in set (0.00 sec)
```

JOIN command

Types of join are;

- Inner join
 - Right join
 - Left join
 - Outer join

RIGHT JOIN

```
select*from customer RIGHT JOIN product on  
product.ProductName=customer. Name;
```

LEFT JOIN

```
select*from customer LEFT JOIN product on  
product.ProductName=customer. Name;
```

C_Id	Name	Address	Email	PostalCode	Product_Id	Model_Number	Product_Type	Price	SerialNumber	ProductName
1	Ahmad	Skp	ahmad01@gmail.com	987	NULL	NULL	NULL	NULL	NULL	NULL
2	Hassan	Lhr	hassan02@gmail.com	8907	NULL	NULL	NULL	NULL	NULL	NULL
3	Zarmeen	Hfz	zarmeen03@gmail.com	5678	NULL	NULL	NULL	NULL	NULL	NULL

INNER JOIN

Some commands for inner join are;

- `select product.Model_no, repairhistory.cost from product INNER JOIN repairhistory on repairhistory.cost = product.Model_no;`
- `select product.Model_no, repairhistory.cost from product INNER JOIN repairhistory on repairhistory.cost > product.Model_no;`
- `select product.Model_no , repairhistory.cost from product INNER JOIN repairhistory on repairhistory.cost < product.Model_no;`

Model_no	cost
MOD-C3	89.99
MOD-B2	89.99
MOD-A1	89.99
MOD-C3	45.50
MOD-B2	45.50
MOD-A1	45.50
MOD-C3	299.00
MOD-B2	299.00
MOD-A1	299.00

SHOW PRIVILEGES

- `SHOW PRIVILEGES;`

Privilege	Context	Comment
Alter	Tables	To alter the table
Alter routine	Functions,Procedures	To alter or drop stored functions/procedures
Create	Databases,Tables,Indexes	To create new databases and tables
Create routine	Databases	To use CREATE FUNCTION/PROCEDURE
Create role	Server Admin	To create new roles
Create temporary tables	Databases	To use CREATE TEMPORARY TABLE
Create view	Tables	To create new views
Create user	Server Admin	To create new users
Delete	Tables	To delete existing rows
Drop	Databases,Tables	To drop databases, tables, and views
Drop role	Server Admin	To drop roles
Event	Server Admin	To create, alter, drop and execute events
Execute	Functions,Procedures	To execute stored routines
File	File access on server	To read and write files on the server
Grant option	Databases,Tables,Functions,Procedures	To give to other users those privileges you possess
Index	Tables	To create or drop indexes
Insert	Tables	To insert data into tables
Lock tables	Databases	To use LOCK TABLES (together with SELECT privilege)
Process	Server Admin	To view the plain text of currently executing queries
Proxy	Server Admin	To make proxy user possible
References	Databases,Tables	To have references on tables
Reload	Server Admin	To reload or refresh tables, logs and privileges
Replication client	Server Admin	To ask where the slave or master servers are
Replication slave	Server Admin	To read binary log events from the master
Select	Tables	To retrieve rows from tables
Show databases	Server Admin	To view all databases with SHOW DATABASES
Show view	Tables	To see views with SHOW CREATE VIEW
Shutdown	Server Admin	To shut down the server
Super	Server Admin	To use KILL thread, SET GLOBAL, CHANGE MASTER, etc.
Trigger	Tables	To use triggers
Create tablespace	Server Admin	To create/alter/drop tablespaces
Update	Tables	To update existing rows
Usage	Server Admin	No privileges - allow connect only
ENCRYPTION_KEY_ADMIN	Server Admin	
INNODB_REDO_LOG_ARCHIVE	Server Admin	
RESOURCE_GROUP_USER	Server Admin	
FIREWALL_EXEMPT	Server Admin	
SET_USER_ID	Server Admin	
SERVICE_CONNECTION_ADMIN	Server Admin	
GROUP_REPLICATION_ADMIN	Server Admin	

- CREATE USER EmanDar IDENTIFIED BY '123456';

USER GRANTS

- CREATE ROLE manager;
- GRANT manager to EmanDar;
- GRANT ALL PRIVILEGES on customer;
- SHOW GRANTS;

```

MySQL 8.0 Command Line Cli  +  -
mysql> CREATE USER EmanDar IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE ROLE manager;
Query OK, 0 rows affected (0.03 sec)

mysql> GRANT manager to EmanDar;
Query OK, 0 rows affected (0.03 sec)

mysql> GRANT ALL PRIVILEGES on customer;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
mysql> SHOW GRANTS;
+-----+
| Grants for root@localhost |
+-----+
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE TABLESPACE, CREATE ROLE, DROP ROLE ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
| GRANT APPLICATION_PASSWORD_ADMIN, AUDIT_ABORT_EXEMPT, AUDIT_ADMIN, AUTHENTICATION_POLICY_ADMIN, BACKUP_ADMIN, BINLOG_ADMIN, BINLOG_ENCRYPTION_ADMIN, CLONE_ADMIN, CONNECTION_ADMIN, ENCRYPTION_KEY_ADMIN, FIREWALL_EXEMPT, FLUSH_OPTIMIZER_COSTS, FLUSH_STATUS, FLUSH_TABLES, FLUSH_USER_RESOURCES, GROUP_REPLICATION_ADMIN, GROUP_RESOURCE_GROUP_ADMIN, RESOURCE_GROUP_USER, ROLE_ADMIN, SENSITIVE_VARIABLES_OBSERVER, SERVICE_CONNECTION_ADMIN, SESSION_VARIABLES_ADMIN, SET_USER_ID, SHOW_ROUTINE_SYSTEM_USER, SYSTEM_VARIABLES_ADMIN, TABLE_ENCRYPTION_ADMIN, TELEMETRY_LOG_ADMIN, XA_RECOVER_ADMIN ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
| GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION
+-----+

```

DELETE table/database

- **Table:** Suppose we have created the table mistakenly we can delete it using following command.

Command : **DROP TABLE user;**

```

MySQL 8.0 Command Line Cli  +  -
the right syntax to use near ', s_description varchar)' at line 1
mysql> CREATE table user(user_id INT PRIMARY KEY, user_name VARCHAR(25), user_address VARCHAR(50));
Query OK, 0 rows affected (0.06 sec)

mysql> desc user;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| user_id | int | NO | PRI | NULL |          |
| user_name | varchar(25) | YES |     | NULL |          |
| user_address | varchar(50) | YES |     | NULL |          |
+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql> DROP user;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
mysql> DROP table user;
Query OK, 0 rows affected (0.04 sec)

mysql> desc user;
ERROR 1146 (42S02): Table 'market.user' doesn't exist

```

- **Database:** Suppose we have created the database mistakenly we can delete it using following command.

Command: **DROP DATABASE market;**

```
mysql> SHOW databases;
+-----+
| Database |
+-----+
| dept    |
| information_schema |
| market   |
| mysql    |
| performance_schema |
| sys      |
+-----+
6 rows in set (0.00 sec)

mysql> DROP database market;
Query OK, 9 rows affected (0.19 sec)

mysql> describe tables;
ERROR 1046 (3D000): No database selected
mysql>
```
