Entity Relationship Diagram

Electronic Product Service System

5/20/25

Database Systems

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)

PROLEM 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 <u>one</u> **warranty.**
2. A **customer** can make <u>many</u> **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.**

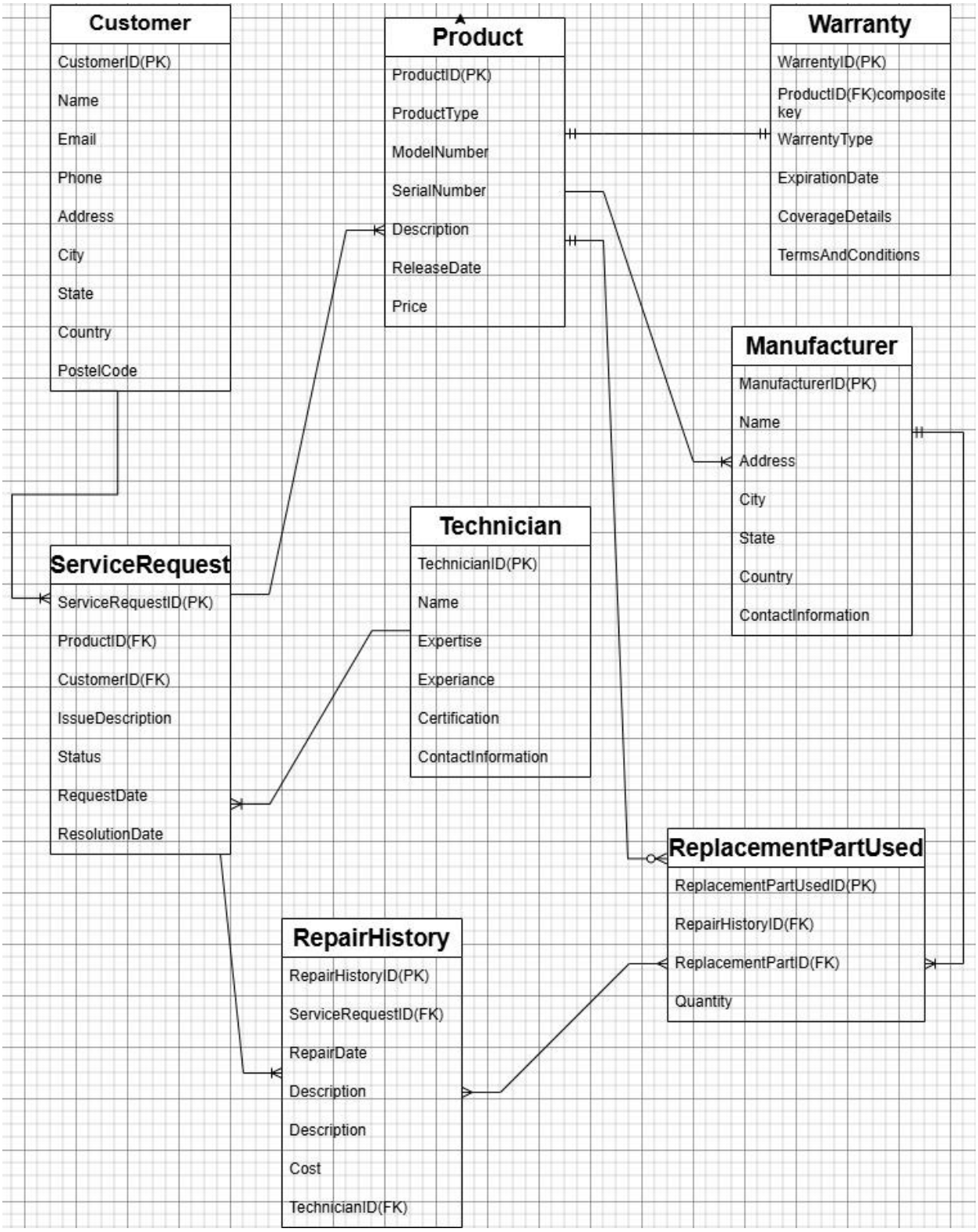
Assumptions

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.

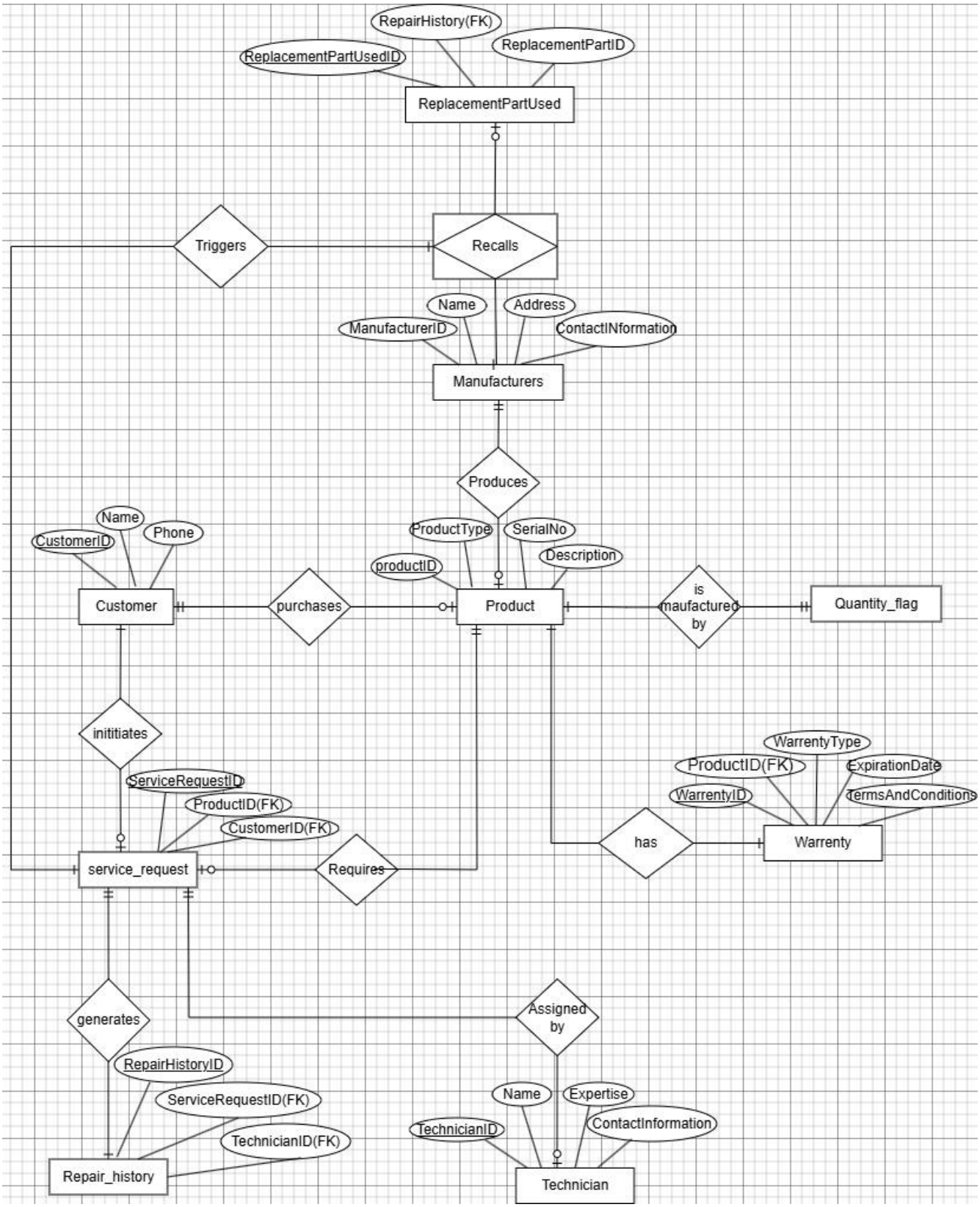
Diagram

ERD.drawio

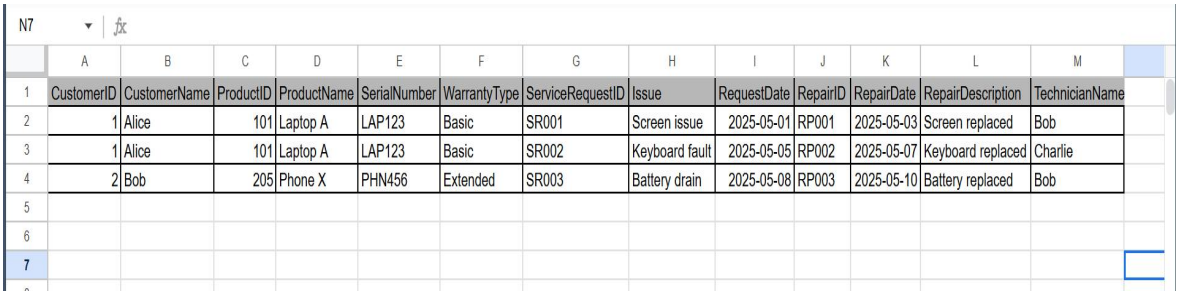
Crow's Foot Model



Chen Model


RepairHistory(FK)
ReplacementPartID
ReplacementPartUsedID
ReplacementPartUsed
Triggers
Recalls
Name
Address
ManufacturerID
ContactINformation
Manufacturers
Produces
Name
CustomerID
Phone
ProductType
SerialNo
Description
productID
Customer
purchases
Product
is maufactured by
Quantity_flag
inititiates
WarrentyType
ProductID(FK)
ExpirationDate
WarrentyID
TermsAndConditions
ServiceRequestID
ProductID(FK)
CustomerID(FK)
has
Warrenty
service_request
Requires
generates
Assigned by
RepairHistoryID
ServiceRequestID(FK)
Name
Expertise
TechnicianID
ContactInformation
TechnicianID(FK)
Repair_history
Technician

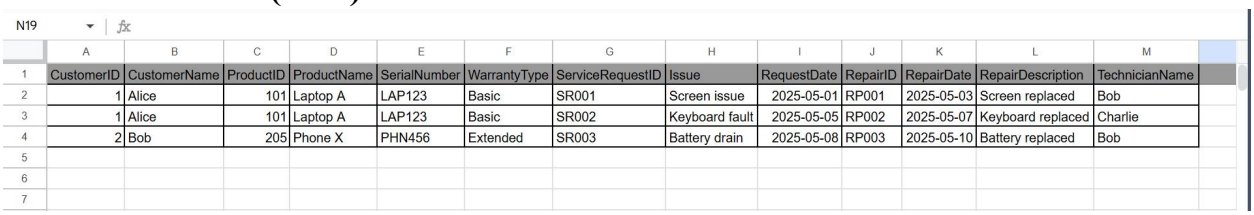
Normalization



	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:

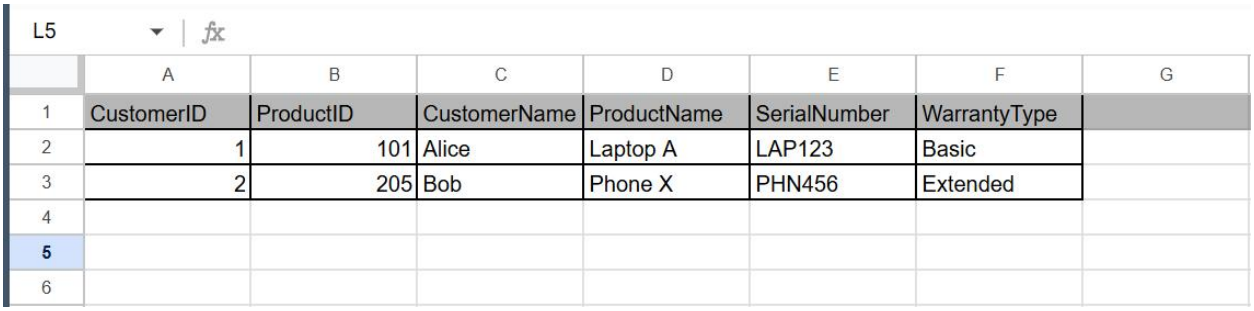
1st Normal Form(1NF):



	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:

H8

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

ServiceRequestIssue Table:

I16

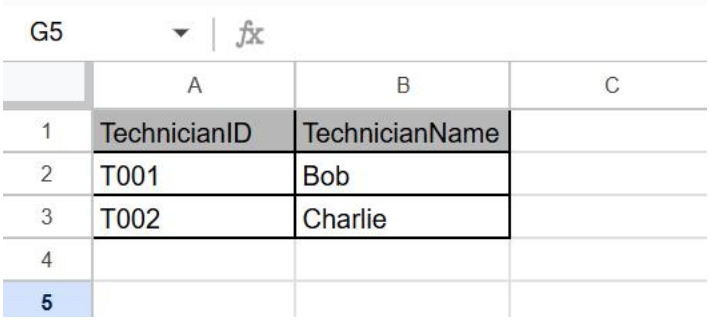
	A	B	C	D	E	F	G
1	ServiceRequestID	CustomerID (FK)	ProductID (FK)	Issue	RequestDate	TechnicianID (FK)	
2	SR001	1	101	Screen issue	2025-05-01	T001	
3	SR002	1	101	Keyboard fault	2025-05-05	T002	
4	SR003	2	205	Battery drain	2025-05-08	T001	
5							
6							

RepairLog Table:

H11

	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:



	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)

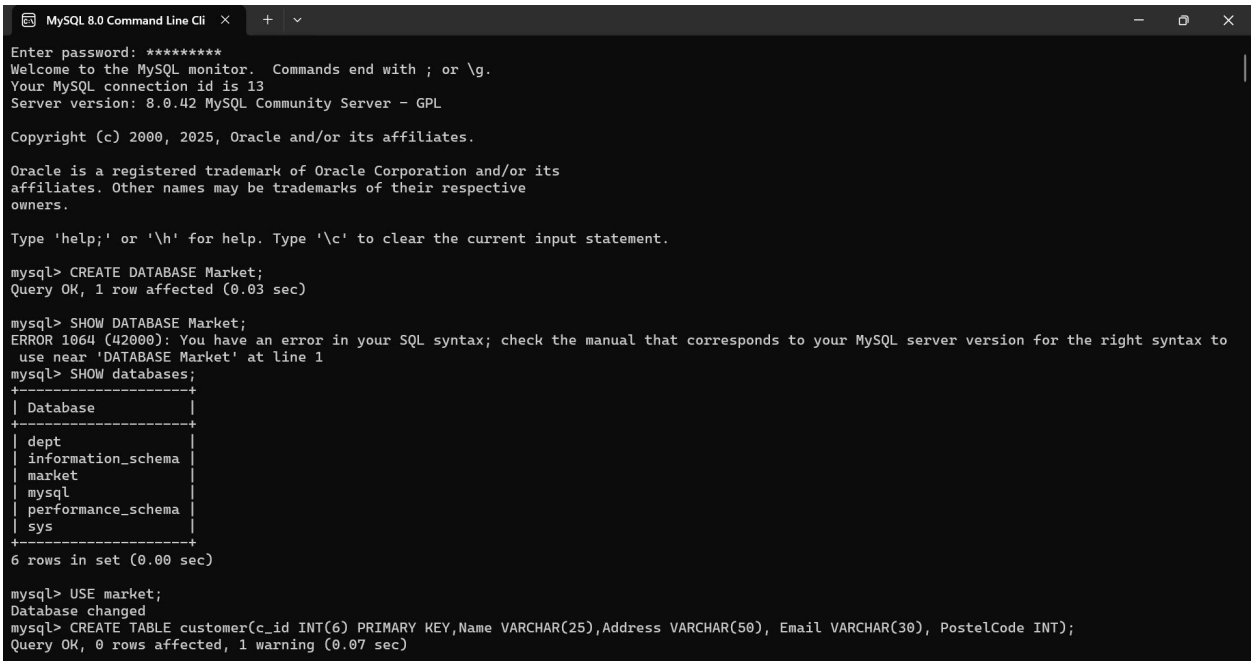
- INSERT INTO
- ADD VALUES IN TABLES
- ADD FOREIGN KEY OR MULTIPLE FOREIGN KEY
- SELECT COMMAND
- UPDATE
- AS
- DSITINCT
- ORDER BY
- WHERE clause
- SELECT COMMAND
- ARITHMETIC OPERATORS
- RELATIONAL OPERATORS
- LOGICAL OPERATORS
- AND BETWEEN
- IN
- LIKE
- NULL and NOT NULL
- AGREGRATE FUNCTIONS
- GROUP 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:

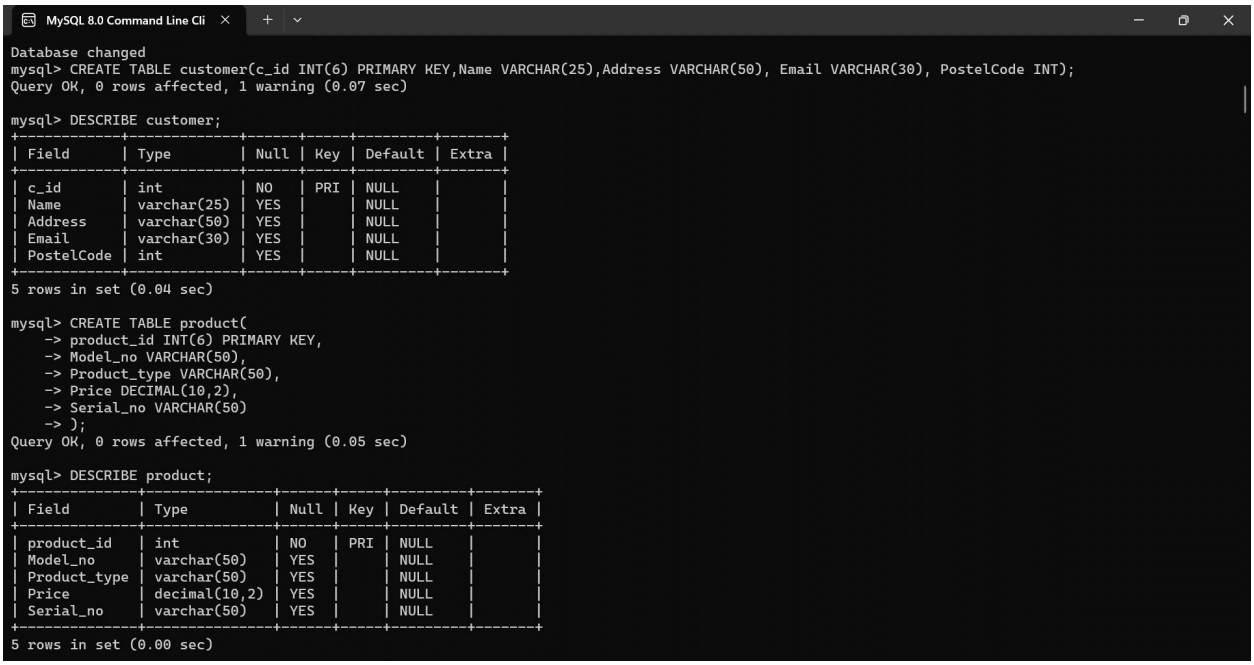
1. Open the SQL Command Line Client.
2. Enter the password.
3. Enter the command to show the databases→ `SHOW DATABASES;`
4. If database is already existing then use command to manipulate in target database.

- `USE database_name;`
- Database name: market;
- Example: `use market;`



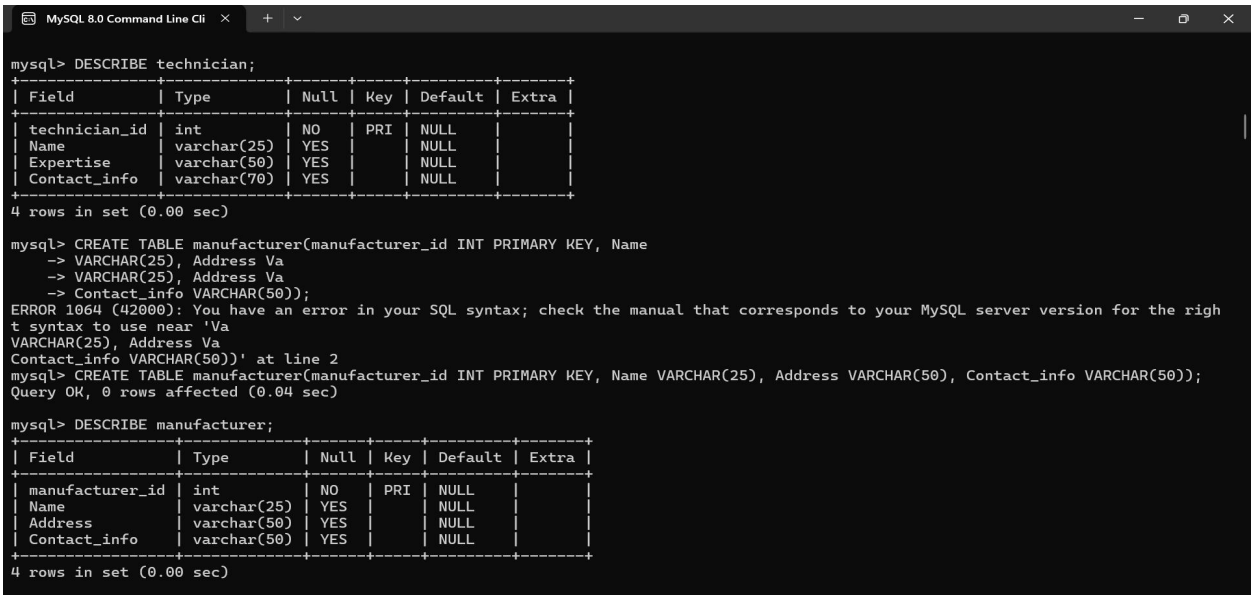
- `Show tables;`
- If you have just created your database so create tables as; `CREATE TABLE table_name();`

- `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

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;

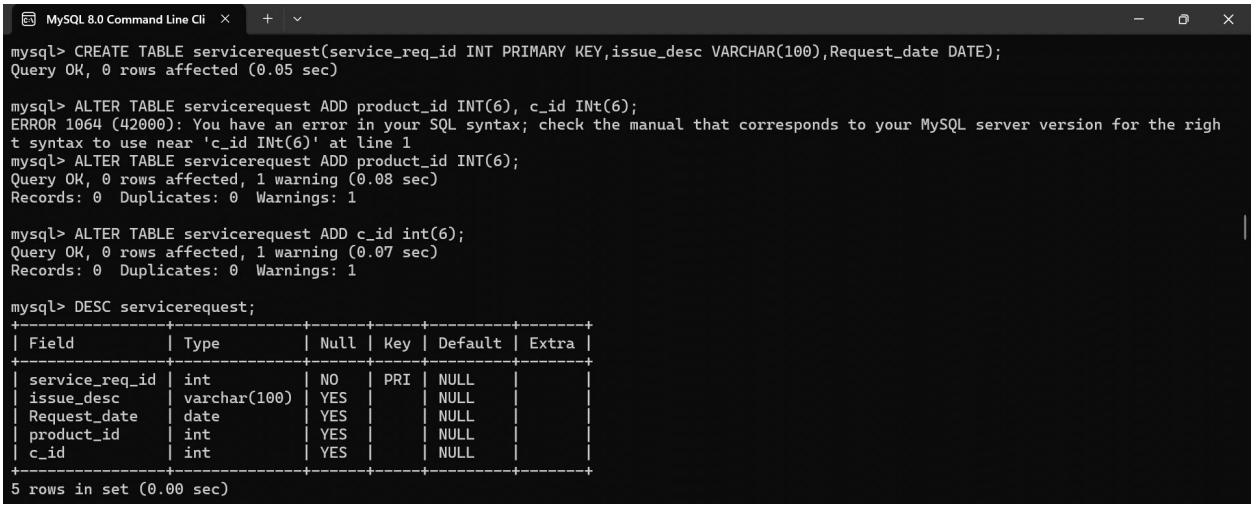


```
MySQL 8.0 Command Line Cli

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 righ
t 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;

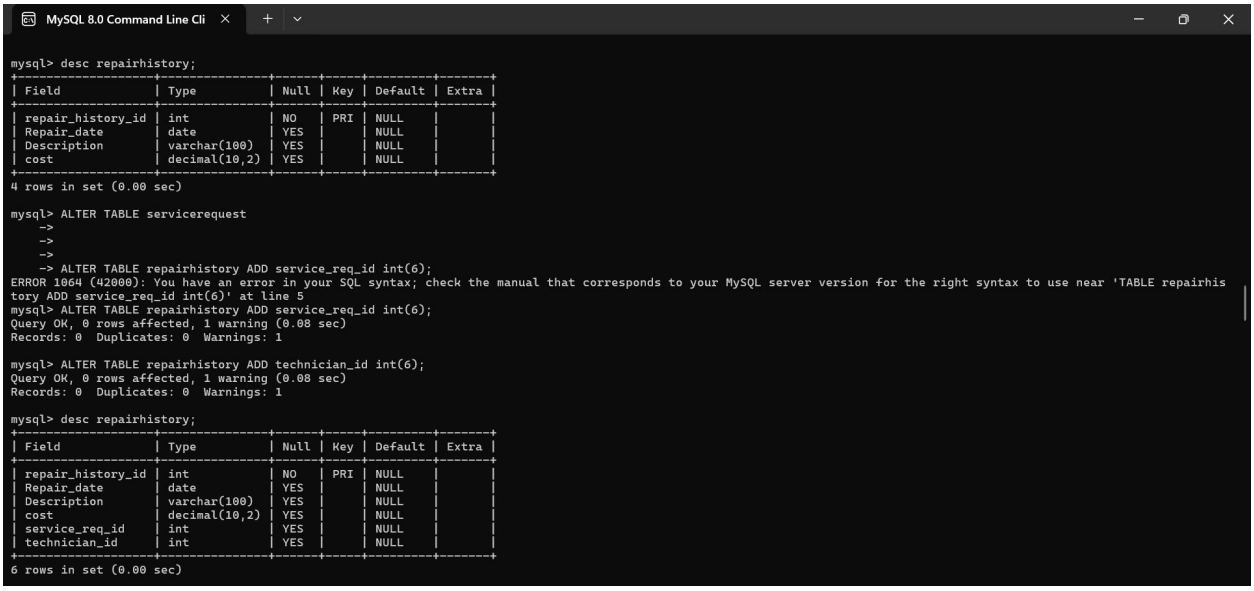
```
MySQL 8.0 Command Line Cli

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 righ
t 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 repairhis
tory 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)
```

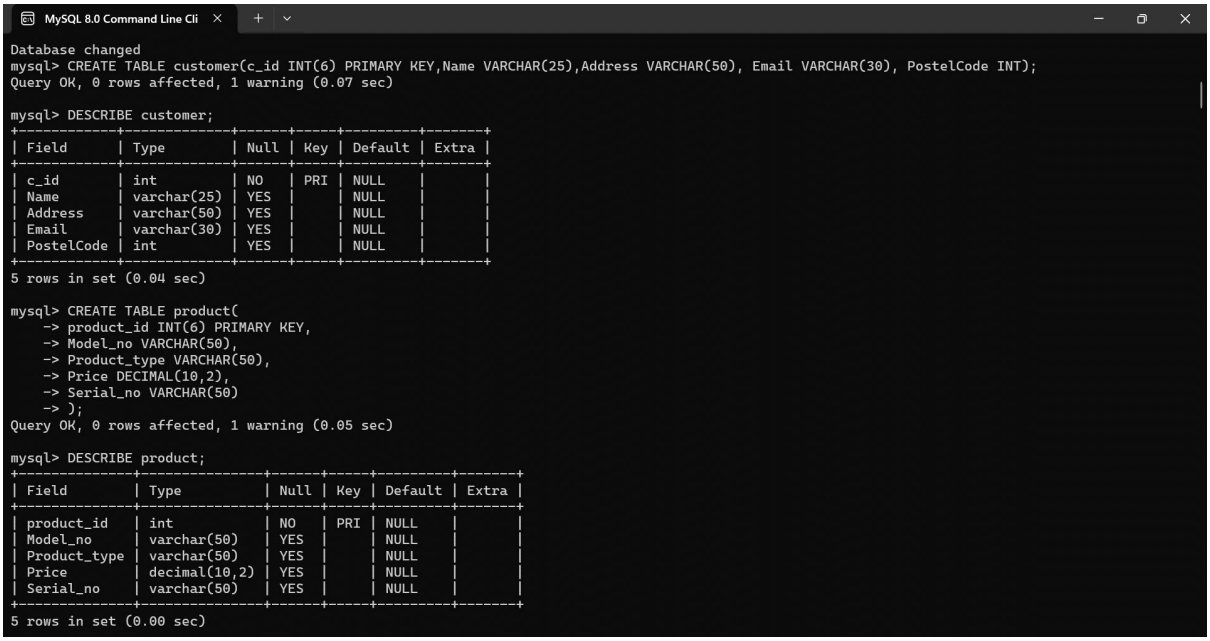
CREATE TABLES

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;`

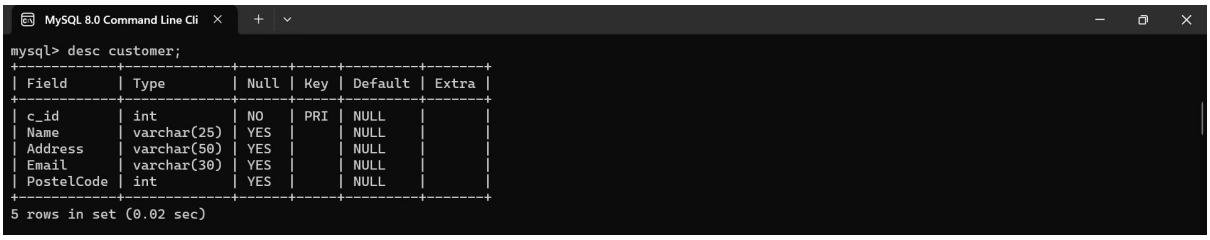


Use of DESCRIBE COMMAND

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

Syntax:

```
DESC table_name;
```



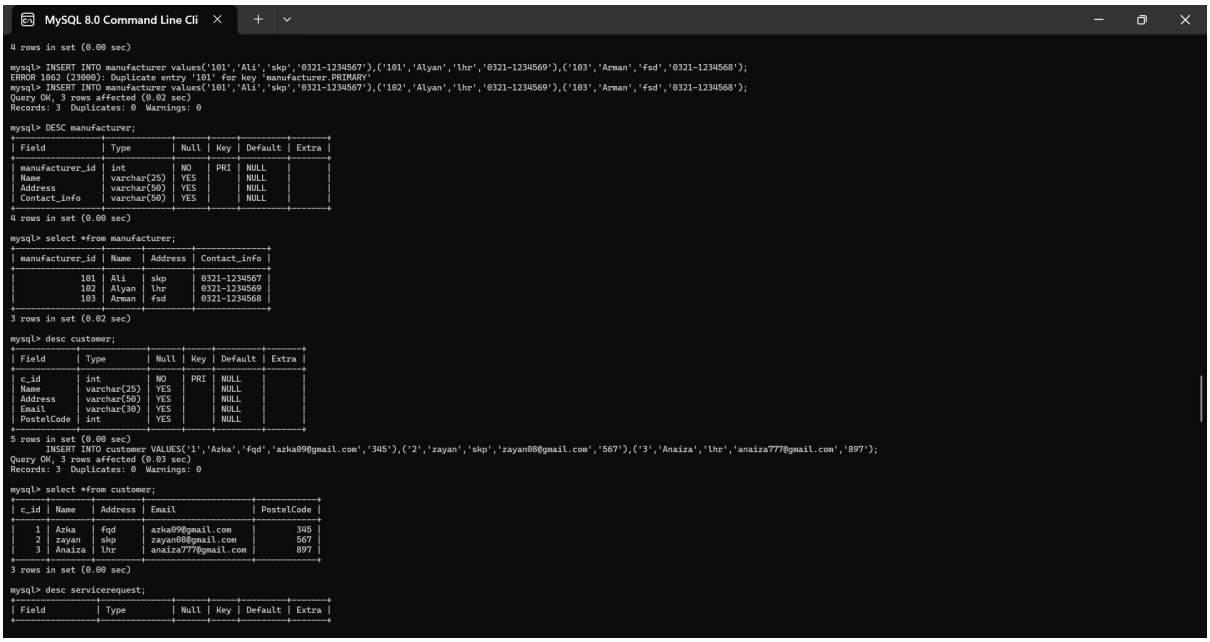
INSERT COMMAD

Syntax;

```
INSERT INTO table_name VALUES (column_1 , VALUES_C……);
```

- Insert values in tables by using command as;

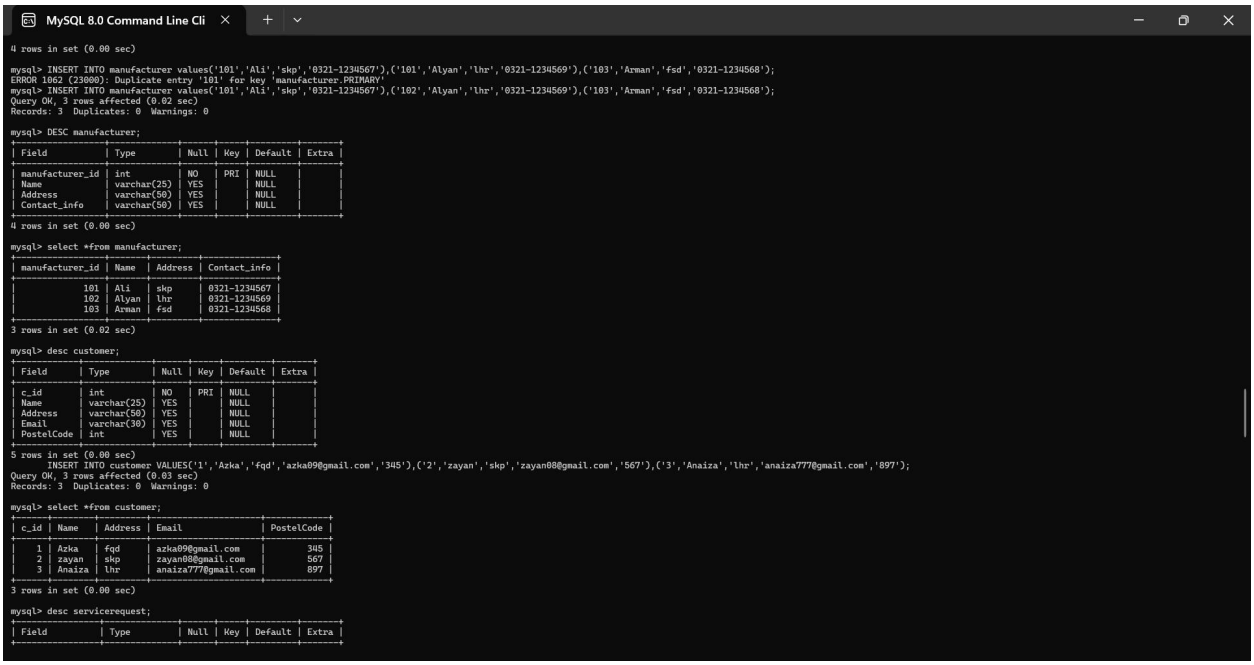
```
INSERT INTO manufacturer values
('101','Ali','skp','0321-1234567'),
('102','Alyan','lhr','0321-1234569'),
('103','Arman','fsd','0321-1234568');
```



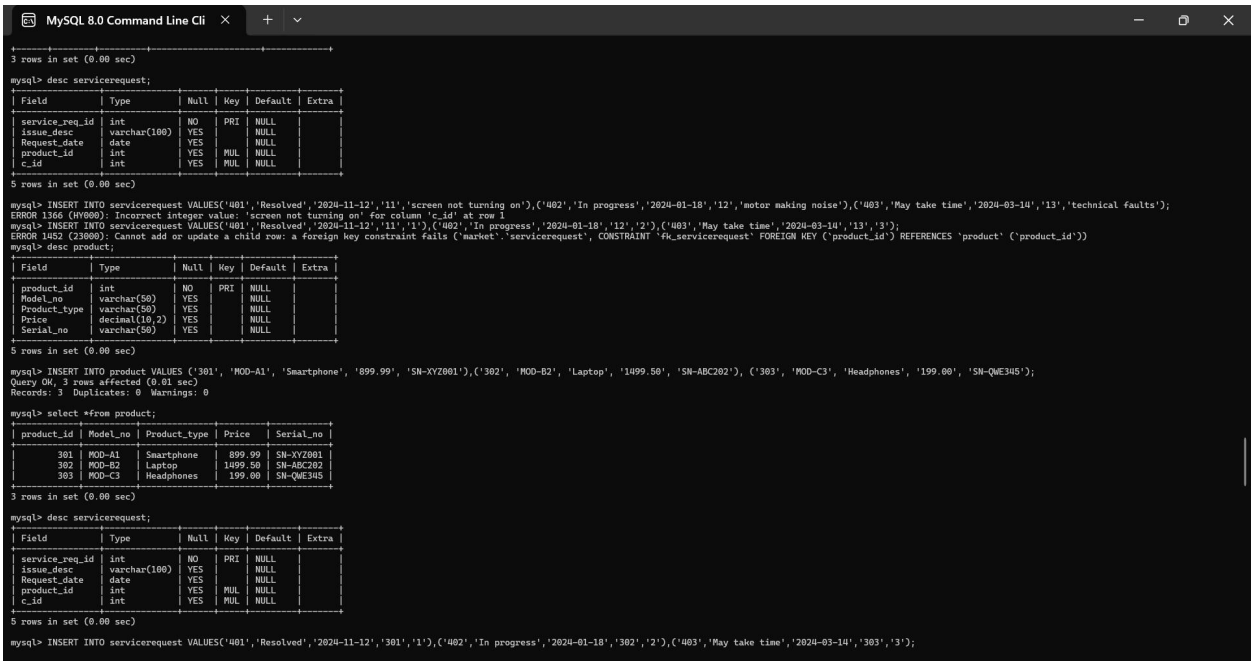
ADD Values in tables

- Insert values in tables by using command as;

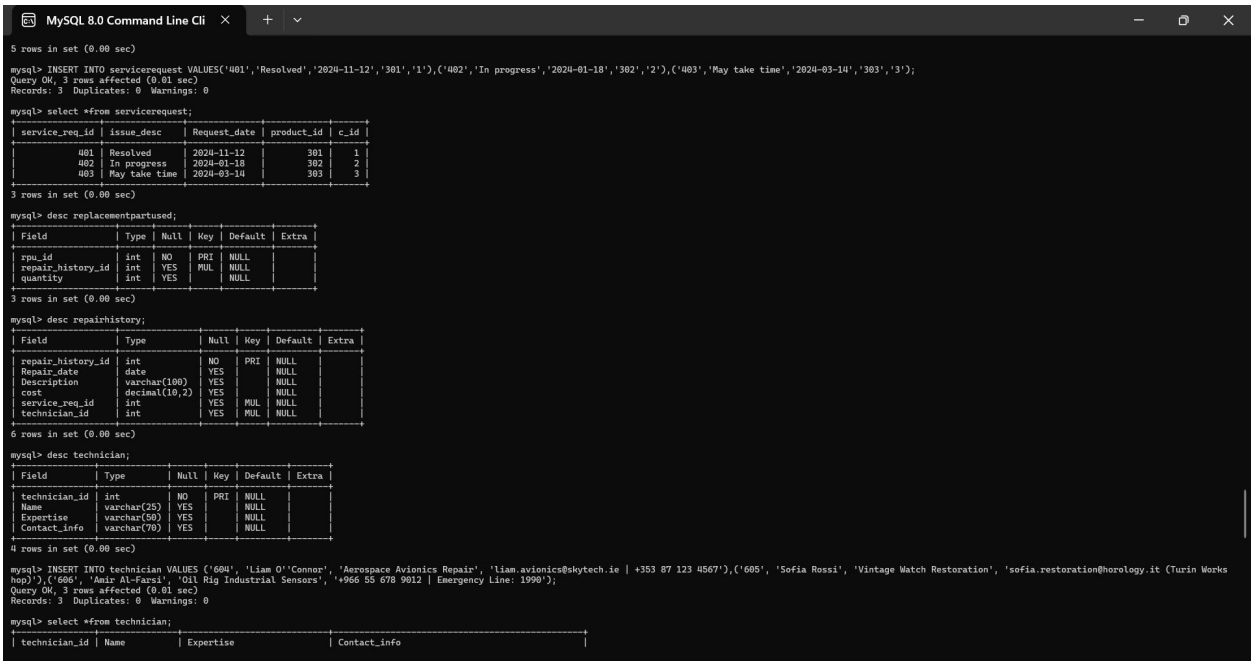
```
INSERT INTO manufacturer values
('101','Ali','skp','0321-1234567'),
('102','Alyan','lhr','0321-1234569'),
('103','Arman','fsd','0321-1234568');
```



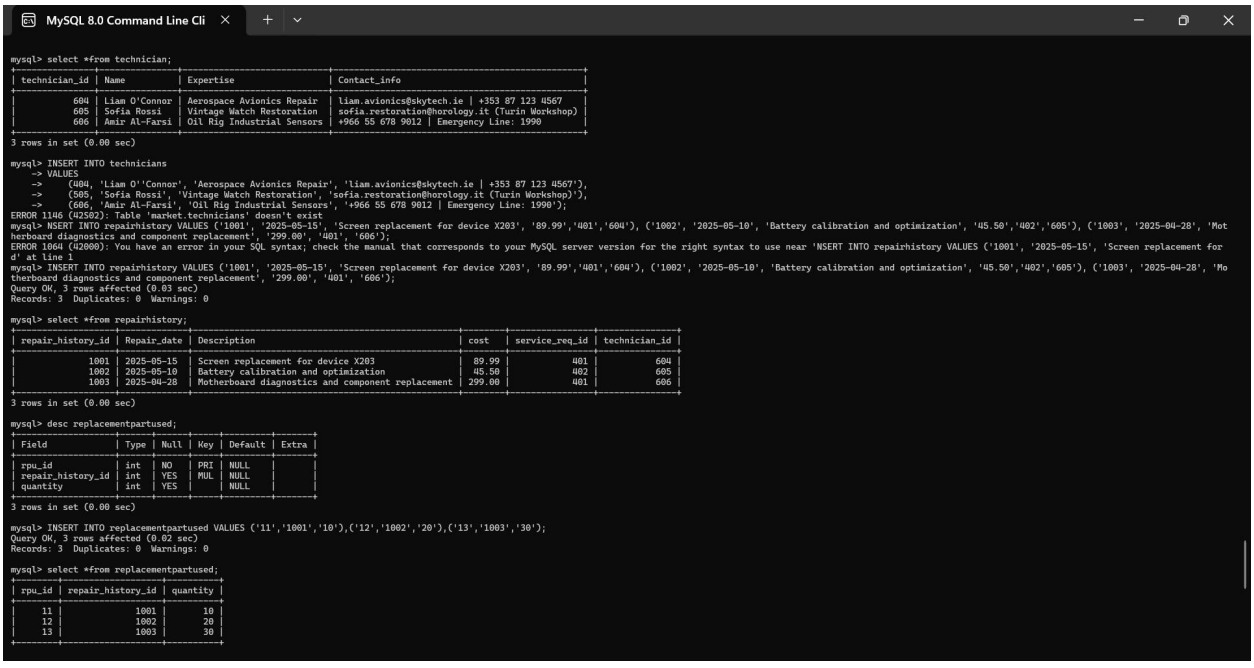
Same as;



Also same for others;



Or



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 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 ;

- 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);
```

- 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 8.0 Command Line Cli

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-XYZ001 |
|        302 | MOD-B2   | Laptop       | 1499.50 | SN-ABC202 |
|        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;
```

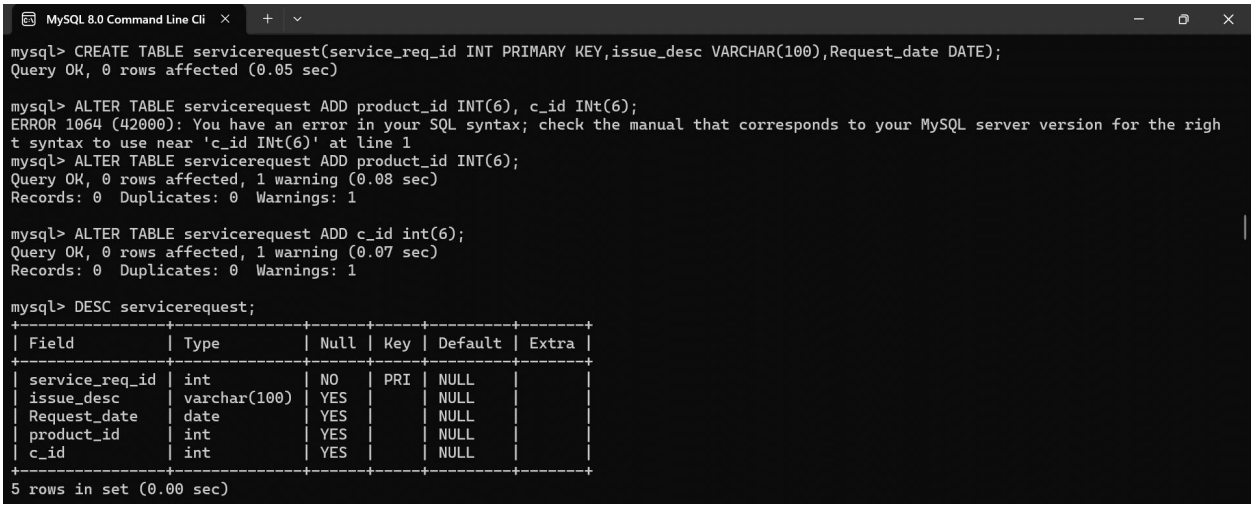
ALTER command

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);`



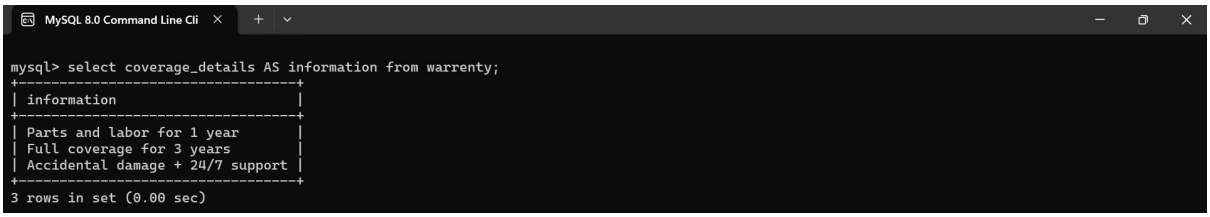
Suppose we want to include the already present table EMPLOYEE the PRIMARY KEY. Then for the sack of that purpose we use:

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;
```



DISTINCT command

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

Syntax:

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

WHERE clause in SQL

Where command is work as such like from the condition .

Syntax:

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

ORDER BY command

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

Syntax:

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

OPERATORS in SQL

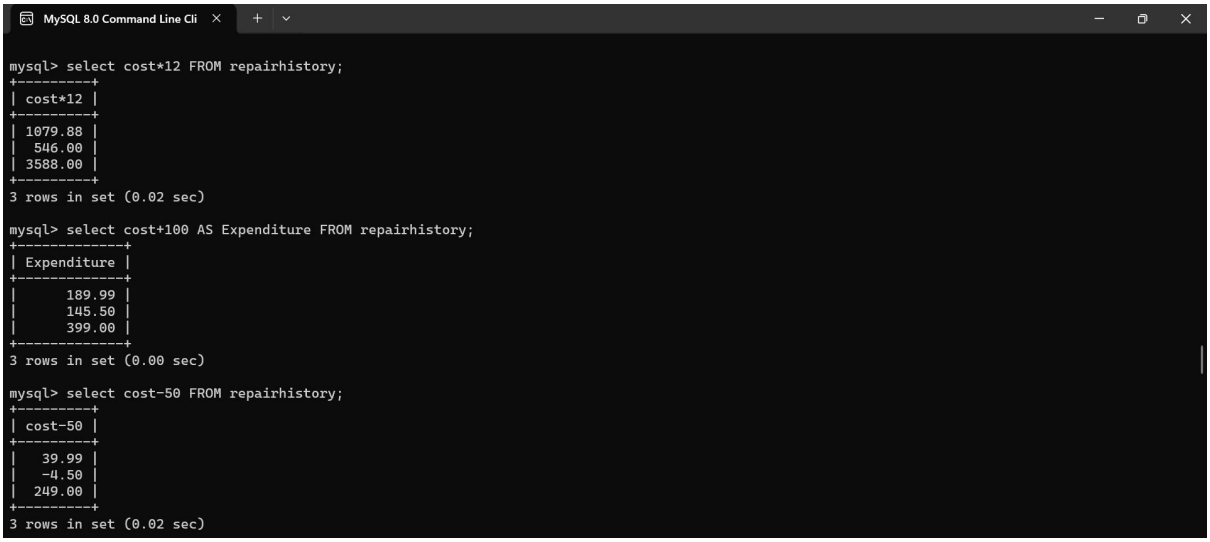
Arithmetic operators including (+,-,*,/).

Syntax:

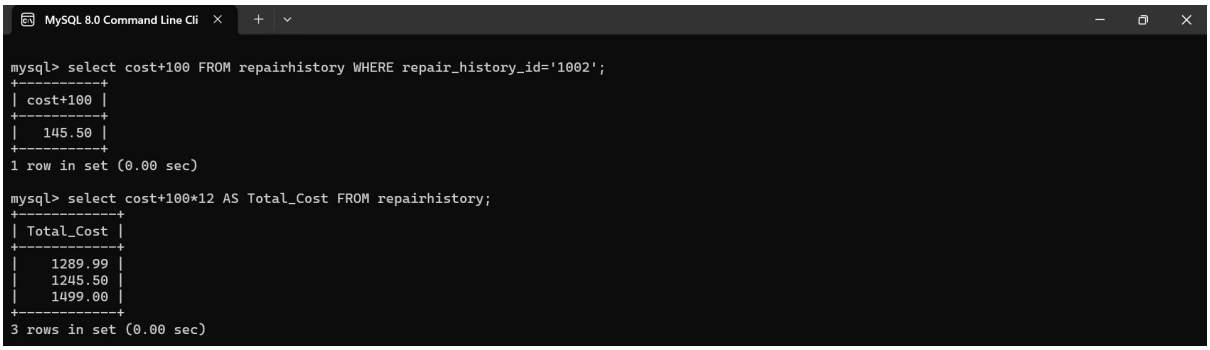
```
SELECT column_name OPERATOR FROM table_name WHERE
column_name='VALUES';
```

Or

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



Or



RELATIONAL OPERATORS

These operators including (>,< ,>=,<=,==,!=).

Syntax:

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

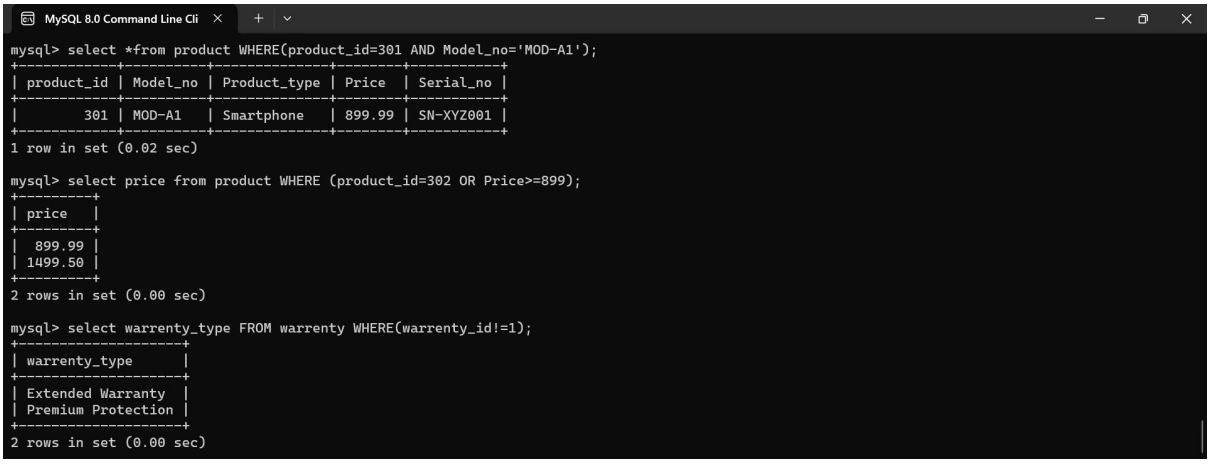
Or

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

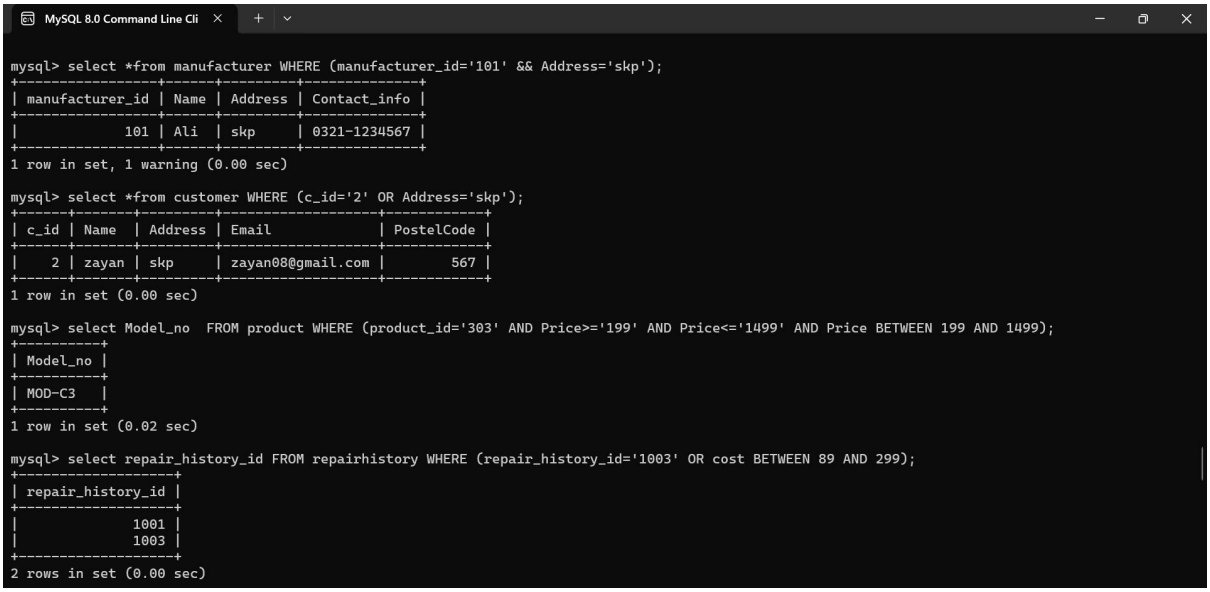
LOGICAL OPERATORS

AND ,OR ,NOT.

SQL QUERIES BY AND,OR OPERATOR;



Use of MULTIPLE AND OPERATOR



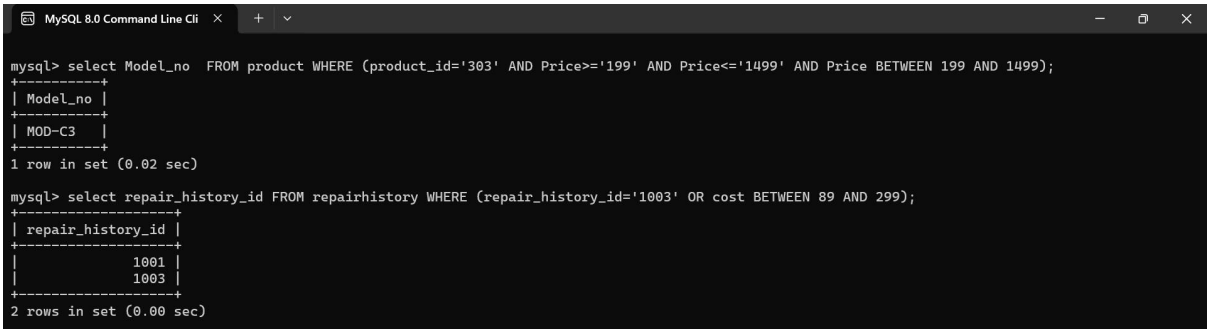
AND BETWEEN

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.



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);
```

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;
```

NULL

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

AGGREGRATION FUNCTIONS:

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

COUNT command

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

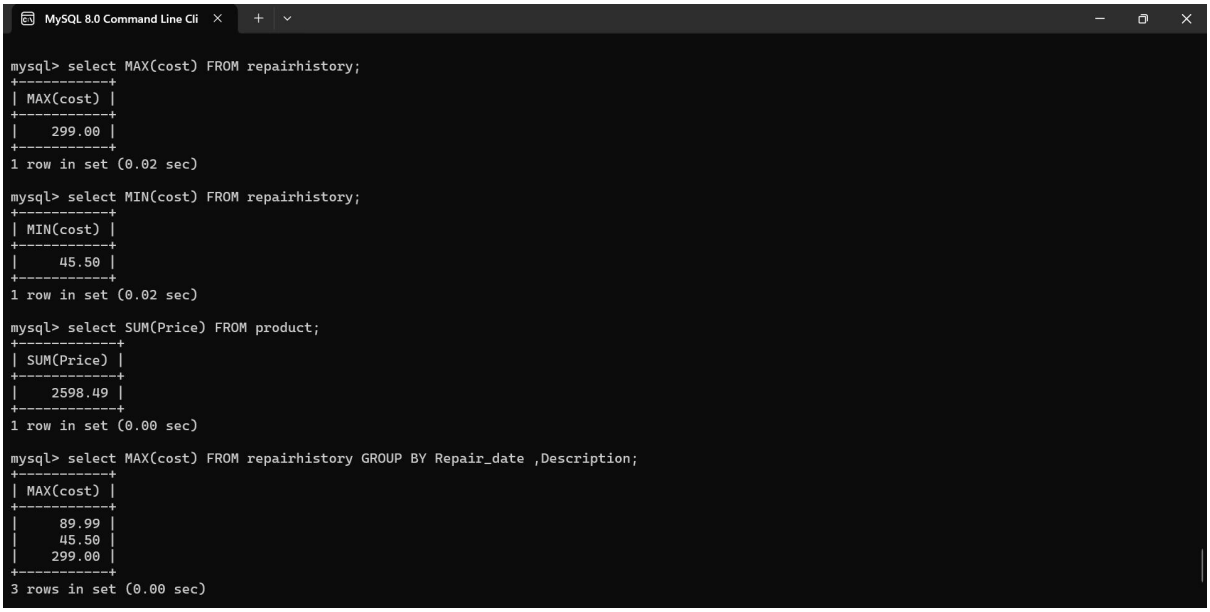
AGGREGATE FUNCTIONS

Syntax:

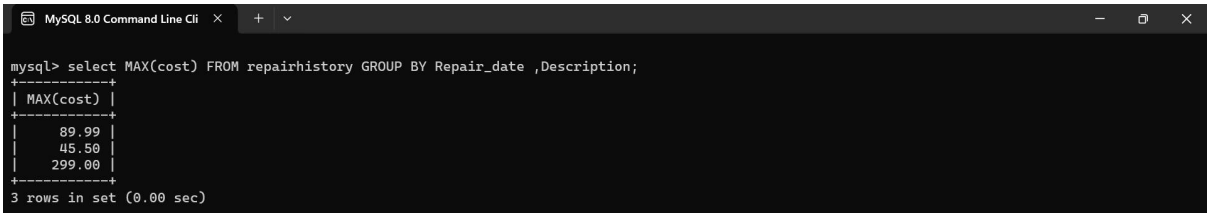
```
SELECT AGGREGATE_FUNCTION(column_name) FROM table_name;
```

AVERAGE

MIN , MAX , SUM



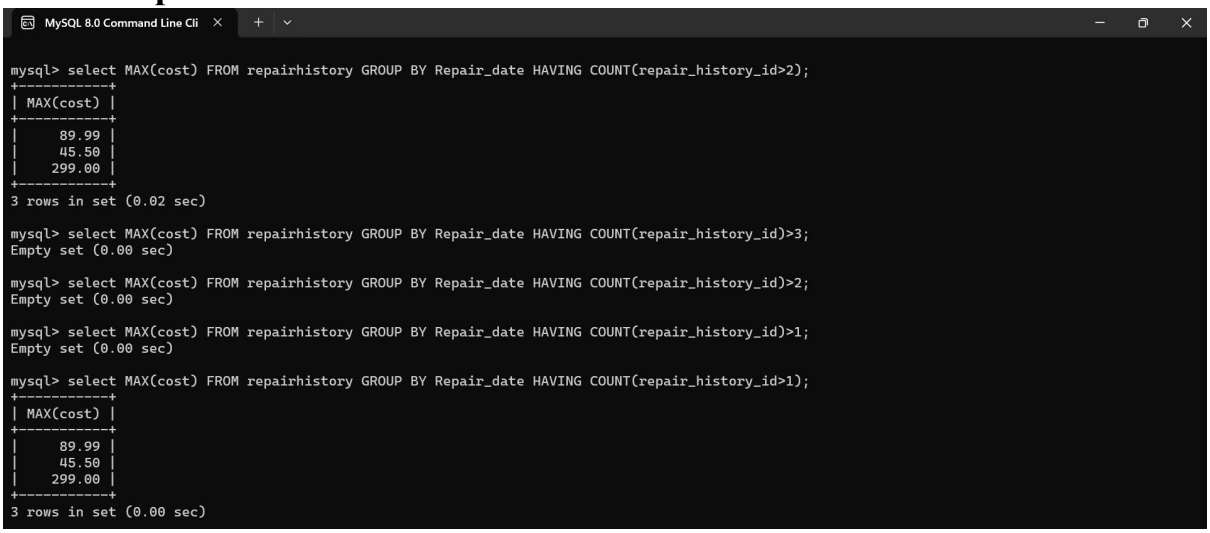
GROUP BY



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 sack of that purpose you have to use the HAVING to filter out of data or attributes as you want from the table.

For example:



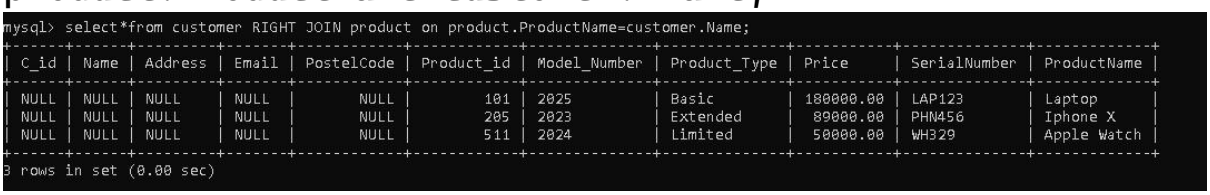
JOIN command

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;
```

```
mysql> select*from customer LEFT JOIN product on product.ProductName=customer.Name;
+------+--------+---------+--------------------+-----------+------------+--------------+--------------+-------+--------------+-------------+
| C_id | Name   | Address | Email              | PostelCode | 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        |
+------+--------+---------+--------------------+-----------+------------+--------------+--------------+-------+--------------+-------------+
3 rows in set (0.00 sec)
```

INNER JOIN

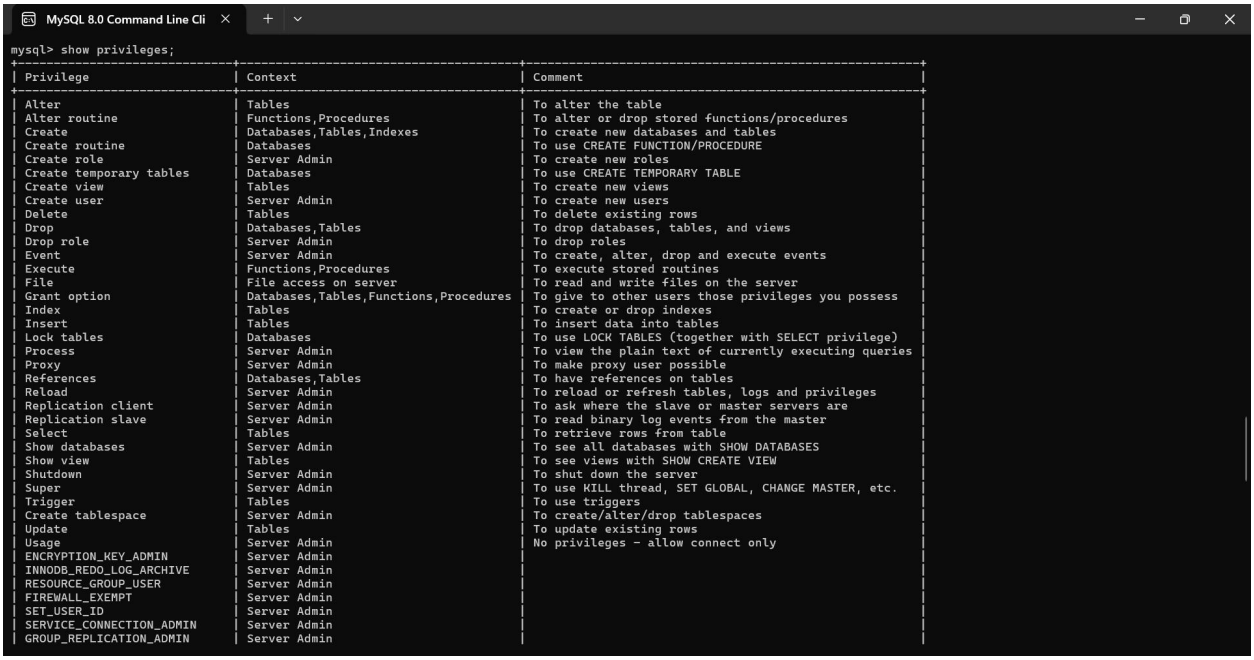
Some commands for inner join are;

- `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;`

```
MySQL 8.0 Command Line Cli

mysql> ^X^C
mysql> select product.Model_no , repairhistory.cost from product INNER JOIN repairhistory on repairhistory.cost < product.Model_no;
Empty set, 9 warnings (0.02 sec)

mysql> 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 |
+----------+--------+
9 rows in set, 9 warnings (0.00 sec)

mysql> select product.Model_no,repairhistory.cost from product INNER JOIN repairhistory on repairhistory.cost = product.Model_no;
Empty set, 3 warnings (0.00 sec)

mysql>
```

SHOW PRIVILEGES

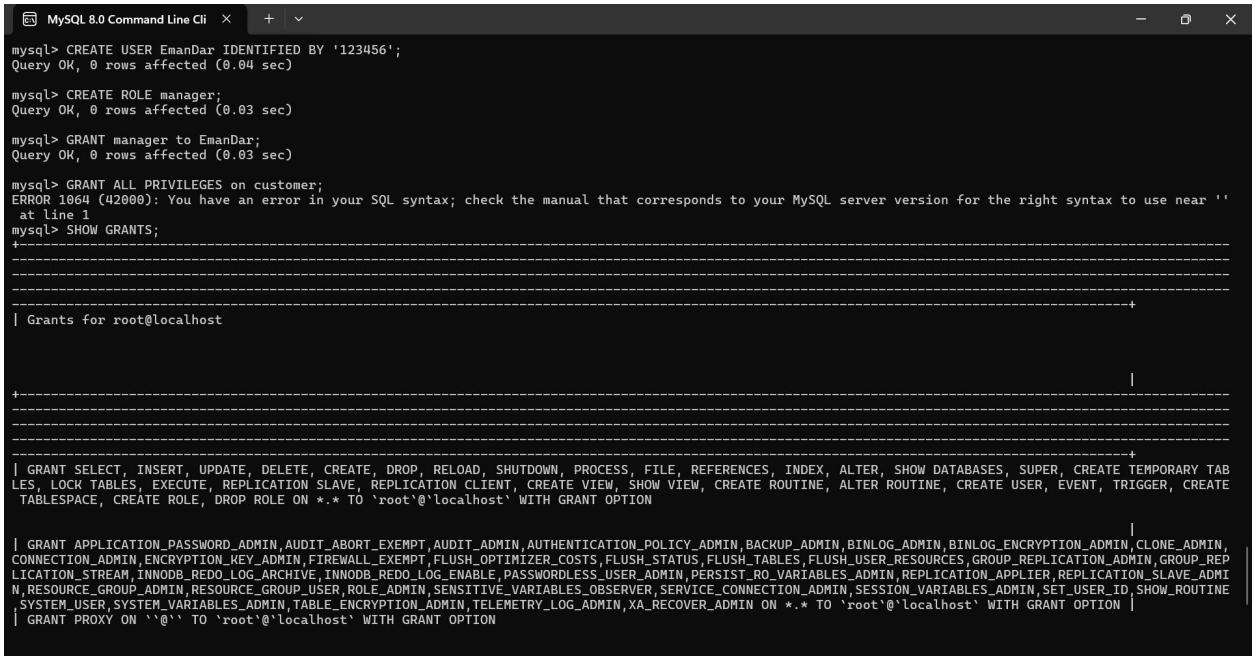
- `SHOW PRIVILEGES;`



- CREATE USER

USER GRANTS

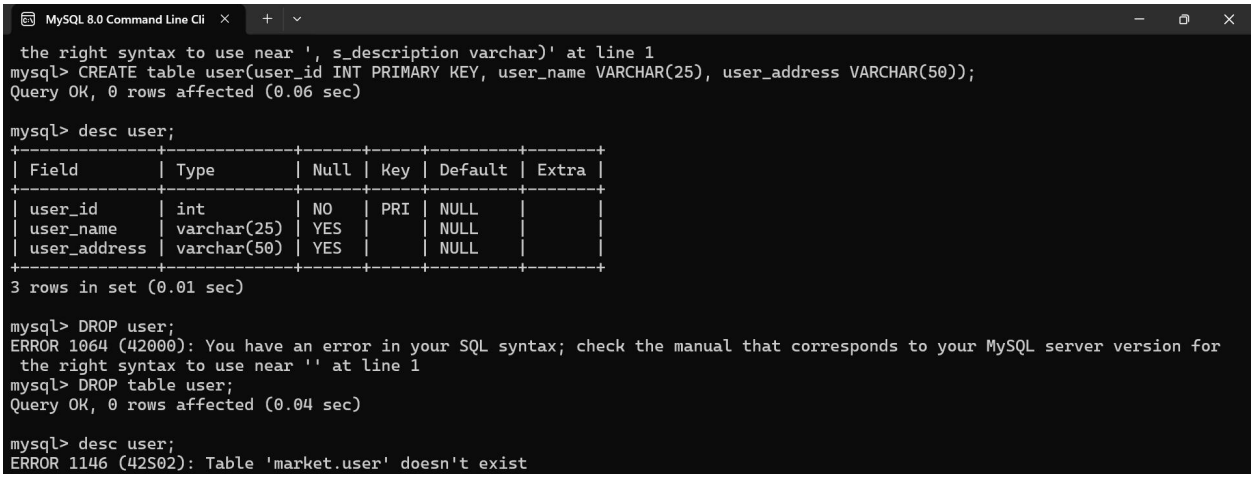
- CREATE USER EmanDar IDENTIFIED BY '123456';
- CREATE ROLE manager;
- GRANT manager to EmanDar;
- GRANT ALL PRIVILEGES on customer;
- SHOW GRANTS;



DELETE table/database

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

Command : `DROP TABLE user;`



- 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>
```
