

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



**DATA BASE MANAGEMENT SYSTEM
LAB REPORT (CIE-2)**

(19CS4PCDBM)

Submitted by

RAVI SAJJANAR (1BM19CS127)

Under the Guidance of

Prof. Sheetal V A

Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Mar-2021 to Jun-2022

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the Lab Assignment work entitled “**Database Management System**” carried out by **RAVI SAJJANAR (1BM19CS127)** who is the bonafide students of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswararajah Technological University, Belgaum during the year 2021-2022. The Lab report has been approved as it satisfies the academic requirements in respect of **Database Management System (19CS4PCDBM) LAB** work prescribed for the said degree.

Signature of the Guide
Prof. Prof. Sheetal VA
Assistant Professor
BMSCE, Bengaluru

Signature of the HOD
Dr. Umadevi V
Associate Prof.& Head, Dept. of CSE
BMSCE, Bengaluru

External Viva

Name of the Examiner

Signature with date

1. _____

2. _____

INDEX

| S.NO | PROGRAM | PAGE |
|------|----------------------------|------|
| 06 | Order Processing Database | 4 |
| 07 | Book dealer Database | 15 |
| 08 | Student Enrolment Database | 21 |
| 09 | Movie Database | 32 |
| 10 | College Database | 43 |

PROGRAM 6: Order Database

Consider the following relations for an Order Processing database application in a company.

CUSTOMER (CUST #: int, cname: String, city: String)

ORDER (order #: int, odate: date, cust #: int, ord-Amt: int)

ITEM (item #: int, unit-price: int)

ORDER-ITEM (order #: int, item #: int, qty: int)

WAREHOUSE (warehouse #: int, city: String)

SHIPMENT (order #: int, warehouse #: int, ship-date: date)

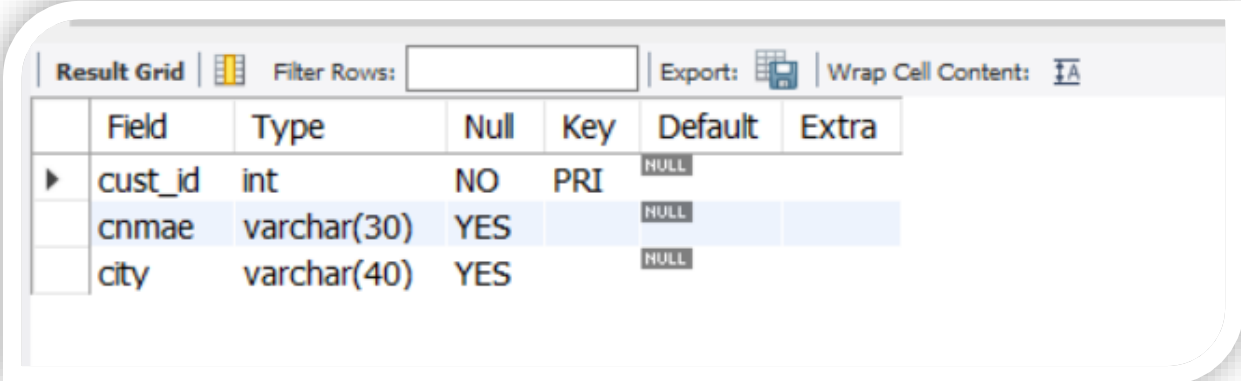
- i) Create the above tables by properly specifying the primary keys and the foreign keys and the foreign keys.
- ii) Enter at least five tuples for each relation.
- iii) Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the middle column is the total numbers of orders by the customer and the last column is the average order amount for that customer.
- iv) List the order# for orders that were shipped from all warehouses that the company has in a specific city.
- v) Demonstrate how you delete item# 10 from the ITEM table and make that field null in the ORDER_ITEM table.

```
create database order_processing;
```

```
/* CUSTOMER Table */
```

```
create table customer(  
    cust_id int,  
    cnmae varchar(30),  
    city varchar(40),  
    primary key(cust_id));
```

```
desc customer;
```





| | Field | Type | Null | Key | Default | Extra |
|---|---------|-------------|------|-----|---------|-------|
| ▶ | cust_id | int | NO | PRI | NULL | |
| | cnmae | varchar(30) | YES | | NULL | |
| | city | varchar(40) | YES | | NULL | |

```
/* ORDERS Table */
```

```
create table orders(  
    order_no int,  
    odate date,  
    cust_id int,  
    ord_amt int,  
    primary key(order_no),  
    foreign key(cust_id)references customer (cust_id));
```

```
desc orders;
```



| Result Grid | | | | | | |
|--|----------|------|------|-----|---------|-------|
| Filter Rows: <input type="text"/> | | | | | | |
| Export:  | | | | | | |
| Wrap Cell Content:  | | | | | | |
| | Field | Type | Null | Key | Default | Extra |
| ▶ | order_no | int | NO | PRI | NULL | |
| | odate | date | YES | | NULL | |
| | cust_id | int | YES | MUL | NULL | |
| | ord_amt | int | YES | | NULL | |

```

/* ITEM Table */
create table item(
    item_id int,
    price int,
    primary key(item_id));

```

```
desc item;
```

| Result Grid | | | | | | |
|--|---------|------|------|-----|---------|-------|
| Filter Rows: <input type="text"/> | | | | | | |
| Export:  | | | | | | |
| Wrap Cell Content:  | | | | | | |
| | Field | Type | Null | Key | Default | Extra |
| ▶ | item_id | int | NO | PRI | NULL | |
| | price | int | YES | | NULL | |

```

/* ORDER_ITEM Table */
create table order_item(
    order_no int,
    item_id int,
    qty int,
    foreign key(order_no)references orders (order_no),
    foreign key(item_id)references item(item_id)on delete set NULL);

```

```
desc order_item;
```

| Result Grid Filter Rows: Export: Wrap Cell Content: | | | | | | |
|---|----------|------|------|-----|---------|-------|
| | Field | Type | Null | Key | Default | Extra |
| ▶ | order_no | int | YES | MUL | NULL | |
| | item_id | int | YES | MUL | NULL | |
| | qty | int | YES | | NULL | |

/* WAREHOUSE Table */

```
create table warehouse(
  warehouse_id int,
  city varchar(40),
  primary key(warehouse_id));
```

desc warehouse;

| Result Grid Filter Rows: Export: Wrap Cell Content: | | | | | | |
|---|--------------|-------------|------|-----|---------|-------|
| | Field | Type | Null | Key | Default | Extra |
| ▶ | warehouse_id | int | NO | PRI | NULL | |
| | city | varchar(40) | YES | | NULL | |

/* SHIPMENT Table */

```
create table shipment(
  order_no int,
  warehouse_id int,
  ship_date date,
  foreign key(order_no)references orders(order_no),
  foreign key(warehouse_id)references warehouse(warehouse_id));
```

desc shipment;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

| | Field | Type | Null | Key | Default | Extra |
|---|--------------|------|------|-----|---------|-------|
| ▶ | order_no | int | YES | MUL | NULL | |
| | warehouse_id | int | YES | MUL | NULL | |
| | ship_date | date | YES | | NULL | |

/* Inserts Values To The CUSTOMER Table */

```
INSERT INTO customer (cust_id, cnmae, city) VALUES (101, 'Asim', 'Gulbarga');
INSERT INTO customer (cust_id, cnmae, city) VALUES (102, 'Amir', 'Delhi');
INSERT INTO customer (cust_id, cnmae, city) VALUES (103, 'Mohsin', 'Mumbai');
INSERT INTO customer (cust_id, cnmae, city) VALUES (104, 'Yaseen', 'Bangalore');
INSERT INTO customer (cust_id, cnmae, city) VALUES (105, 'Aejaz', 'Chennai');
```

select * from customer;

Result Grid

Filter Rows:

Edit:

Export/Import:

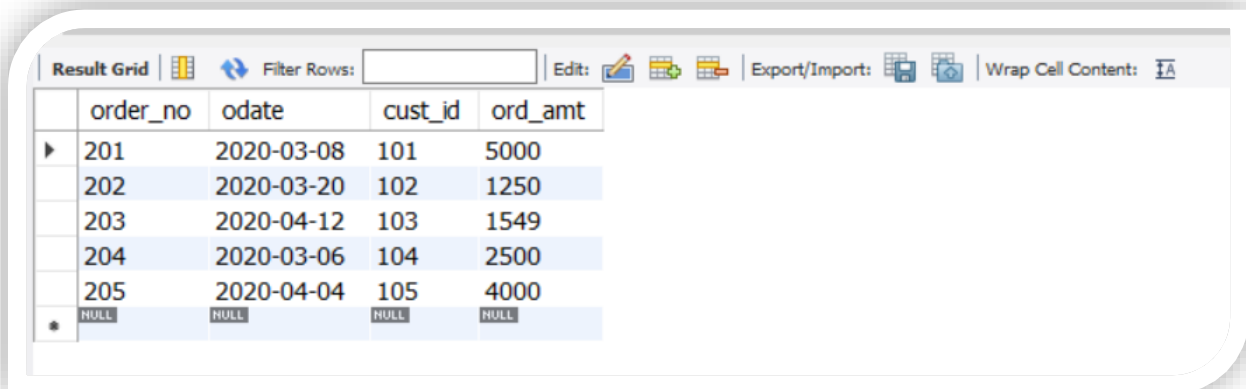
Wrap Cell Content:

| | cust_id | cnmae | city |
|---|---------|--------|-----------|
| ▶ | 101 | Asim | Gulbarga |
| | 102 | Amir | Delhi |
| | 103 | Mohsin | Mumbai |
| | 104 | Yaseen | Bangalore |
| | 105 | Aejaz | Chennai |
| * | NULL | NULL | NULL |

/* Inserts Values To The ORDERS Table */

```
INSERT INTO orders (order_no, odate, cust_id, ord_amt) VALUES ('201', '2020-03-08', '101', '5000');
INSERT INTO orders (order_no, odate, cust_id, ord_amt) VALUES ('202', '2020-03-20', '102', '1250');
INSERT INTO orders (order_no, odate, cust_id, ord_amt) VALUES ('203', '2020-04-12', '103', '1549');
INSERT INTO orders (order_no, odate, cust_id, ord_amt) VALUES ('204', '2020-03-06', '104', '2500');
INSERT INTO orders (order_no, odate, cust_id, ord_amt) VALUES ('205', '2020-04-04', '105', '4000');
```


select * from orders;



| | order_no | odate | cust_id | ord_amt |
|---|----------|------------|---------|---------|
| ▶ | 201 | 2020-03-08 | 101 | 5000 |
| | 202 | 2020-03-20 | 102 | 1250 |
| | 203 | 2020-04-12 | 103 | 1549 |
| | 204 | 2020-03-06 | 104 | 2500 |
| | 205 | 2020-04-04 | 105 | 4000 |
| * | NULL | NULL | NULL | NULL |

/* Inserts Values To The ITEM Table */

INSERT INTO item (item_id, price) VALUES ('301', '2500');

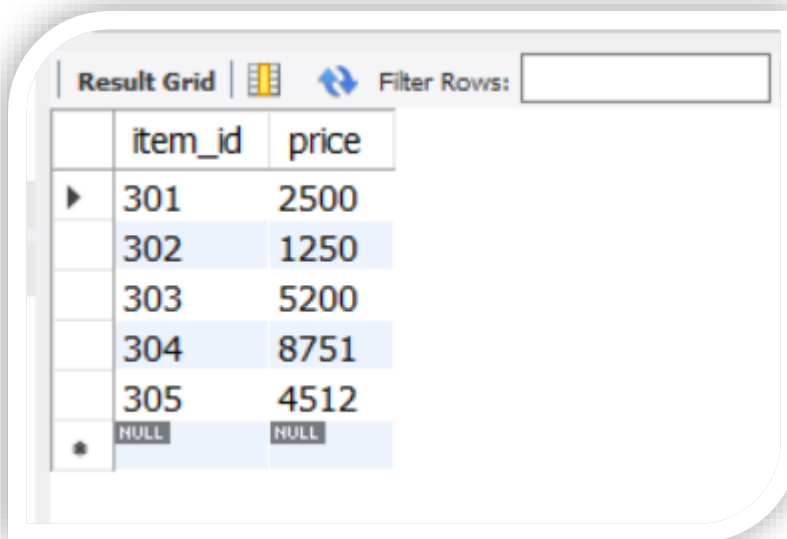
INSERT INTO item (item_id, price) VALUES ('302', '1250');

INSERT INTO item (item_id, price) VALUES ('303', '5200');

INSERT INTO item (item_id, price) VALUES ('304', '8751');

INSERT INTO item (item_id, price) VALUES ('305', '4512');

select * from item;



| | item_id | price |
|---|---------|-------|
| ▶ | 301 | 2500 |
| | 302 | 1250 |
| | 303 | 5200 |
| | 304 | 8751 |
| | 305 | 4512 |
| * | NULL | NULL |

/* Inserts Values To The ORDER_ITEM Table */

INSERT INTO order_item (order_no, item_id, qty) VALUES ('201', '301', '2');

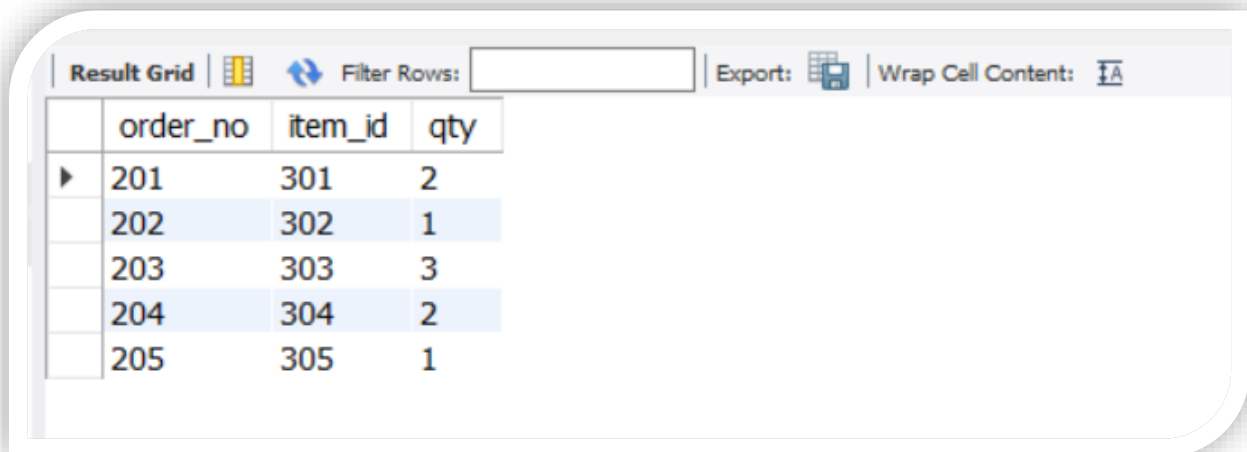
INSERT INTO order_item (order_no, item_id, qty) VALUES ('202', '302', '1');

INSERT INTO order_item (order_no, item_id, qty) VALUES ('203', '303', '3');

INSERT INTO order_item (order_no, item_id, qty) VALUES ('204', '304', '2');

```
INSERT INTO order_item (order_no, item_id, qty) VALUES ('205', '305', '1');
```

```
select * from order_item;
```



| | order_no | item_id | qty |
|---|----------|---------|-----|
| ▶ | 201 | 301 | 2 |
| | 202 | 302 | 1 |
| | 203 | 303 | 3 |
| | 204 | 304 | 2 |
| | 205 | 305 | 1 |

```
/* Inserts Values To The WAREHOUSE Table */
```

```
INSERT INTO warehouse (warehouse_id, city) VALUES ('501', 'Bangalore');
```

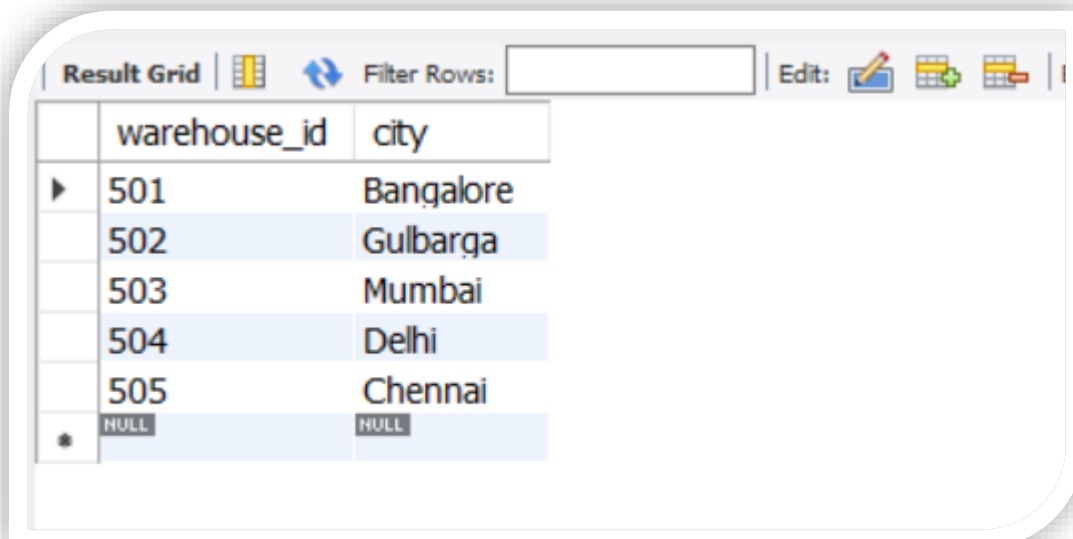
```
INSERT INTO warehouse (warehouse_id, city) VALUES ('502', 'Gulbarga');
```

```
INSERT INTO warehouse (warehouse_id, city) VALUES ('503', 'Mumbai');
```

```
INSERT INTO warehouse (warehouse_id, city) VALUES ('504', 'Delhi');
```

```
INSERT INTO warehouse (warehouse_id, city) VALUES ('505', 'Chennai');
```

```
select * from warehouse;
```



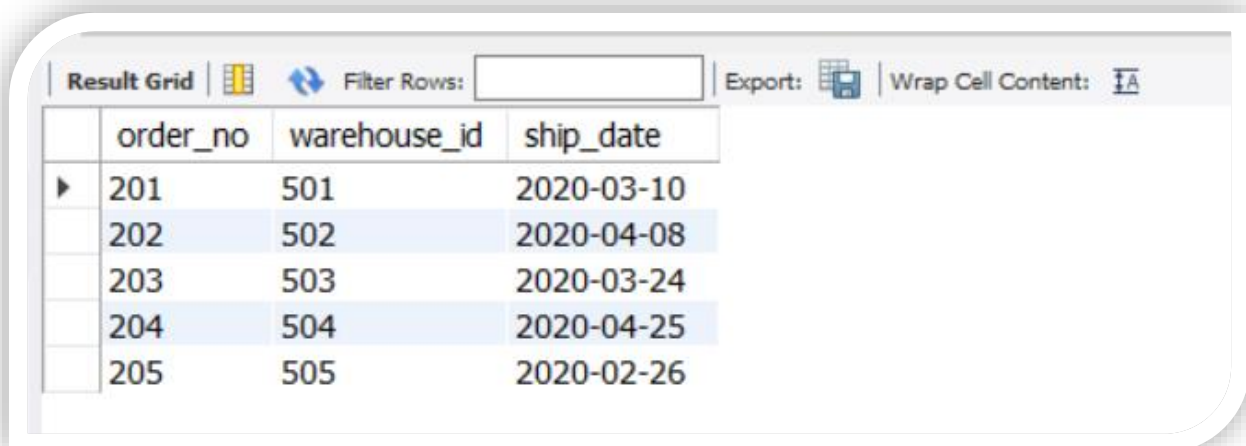
| | warehouse_id | city |
|---|--------------|-----------|
| ▶ | 501 | Bangalore |
| | 502 | Gulbarga |
| | 503 | Mumbai |
| | 504 | Delhi |
| | 505 | Chennai |
| * | NULL | NULL |

```

/* Inserts Values To The SHIPMENT Table */
INSERT INTO shipment (order_no, warehouse_id, ship_date) VALUES ('201', '501',
'2020-03-10');
INSERT INTO shipment (order_no, warehouse_id, ship_date) VALUES ('202', '502',
'2020-04-08');
INSERT INTO shipment (order_no, warehouse_id, ship_date) VALUES ('203', '503',
'2020-03-24');
INSERT INTO shipment (order_no, warehouse_id, ship_date) VALUES ('204', '504',
'2020-04-25');
INSERT INTO shipment (order_no, warehouse_id, ship_date) VALUES ('205', '505',
'2020-02-26');

```

```
select * from shipment;
```



The screenshot shows a database interface with a 'Result Grid' tab. It displays the results of a SQL query. The grid has columns for 'order_no', 'warehouse_id', and 'ship_date'. There are 5 rows of data, each with a blue arrow icon in the first column. The interface also includes a 'Filter Rows' search bar, an 'Export' button, and a 'Wrap Cell Content' toggle.

| | order_no | warehouse_id | ship_date |
|---|----------|--------------|------------|
| ▶ | 201 | 501 | 2020-03-10 |
| | 202 | 502 | 2020-04-08 |
| | 203 | 503 | 2020-03-24 |
| | 204 | 504 | 2020-04-25 |
| | 205 | 505 | 2020-02-26 |

```

/* 01. Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the
middle column is the total numbers of orders by the customer and the last column is the
average order amount for that customer. */

```

```

select C.cnmae,count(order_no) as No_Of_Orders,avg(ord_amt) as Avg_Amount
from customer C,orders O
where C.cust_id = O.cust_id
group by C.cnmae;

```

| Result Grid | | | |
|-------------|--------|--------------------|------------|
| | | Filter Rows: | |
| | | Export: | |
| | | Wrap Cell Content: | |
| | cnmae | No_Of_Orders | Avg_Amount |
| ▶ | Asim | 1 | 5000.0000 |
| | Amir | 1 | 1250.0000 |
| | Mohsin | 1 | 1549.0000 |
| | Yaseen | 1 | 2500.0000 |
| | Aejaz | 1 | 4000.0000 |

```

/* 02. List the order# for orders that were shipped from all warehouses that the company
has in a specific city. */
select order_no from orders where order_no in
(select order_no from shipment
where warehouse_id in
(select warehouse_id from warehouse
where city = 'Bangalore'));

```

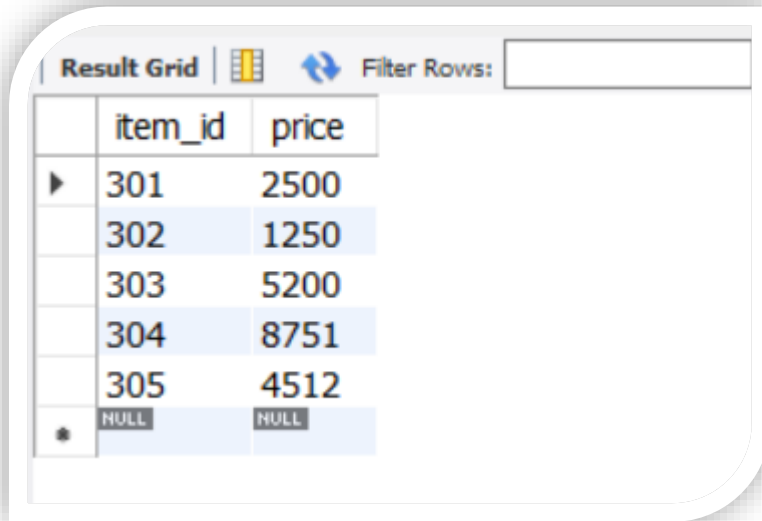
| Result Grid | |
|--------------|----------|
| Filter Rows: | |
| Edit: | |
| | order_no |
| ▶ | 201 |
| * | NULL |

```

/* 03. Demonstrate how you delete item# 10 from the ITEM table and make that field
null in the ORDER_ITEM table. */

```

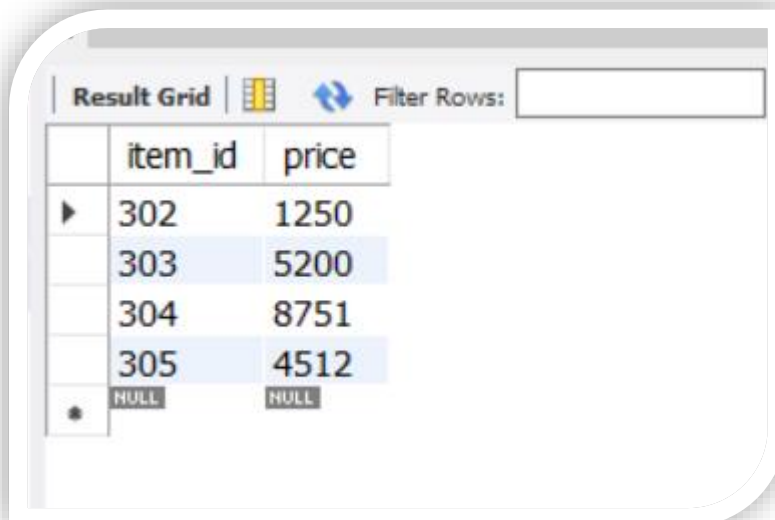
```
select * from item;
```



The screenshot shows a database interface with a 'Result Grid' tab. It contains a table with two columns: 'item_id' and 'price'. The table has six rows: five data rows and one row with NULL values. The first data row (item_id 301) is highlighted with a blue background. To the left of the table is a vertical list of row numbers 1 through 6, with a small triangle icon next to row 1. Above the table is a 'Filter Rows:' input field.

| | item_id | price |
|---|---------|-------|
| 1 | 301 | 2500 |
| 2 | 302 | 1250 |
| 3 | 303 | 5200 |
| 4 | 304 | 8751 |
| 5 | 305 | 4512 |
| 6 | NULL | NULL |

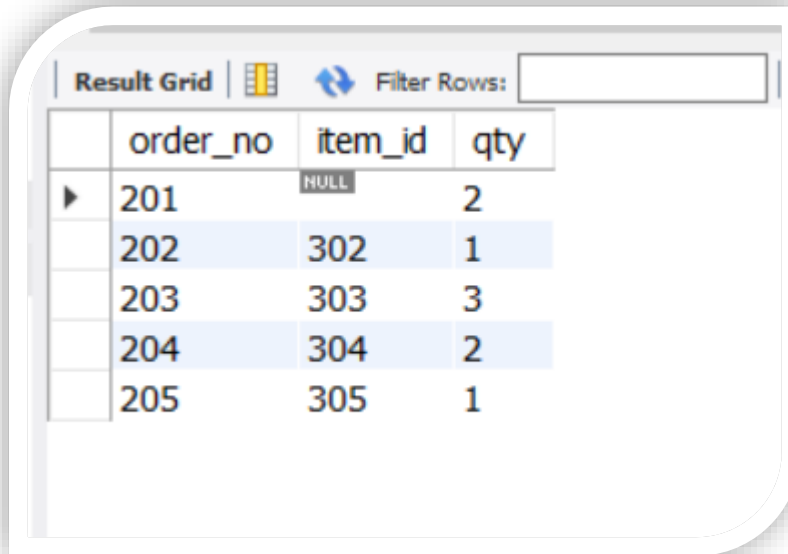
```
delete from item where item_id = 301;  
select * from item;
```



The screenshot shows the same database interface as before, but the first row (item_id 301) has been removed. The table now has five data rows and one row with NULL values. The first data row (item_id 302) is highlighted with a blue background. The vertical list of row numbers on the left now ranges from 1 to 5. The 'Filter Rows:' input field remains empty.

| | item_id | price |
|---|---------|-------|
| 1 | 302 | 1250 |
| 2 | 303 | 5200 |
| 3 | 304 | 8751 |
| 4 | 305 | 4512 |
| 5 | NULL | NULL |

```
select * from order_item;
```



The image shows a screenshot of a database application's 'Result Grid'. At the top, there is a tab labeled 'Result Grid' and a 'Filter Rows:' input field. Below this is a table with four columns: 'order_no', 'item_id', and 'qty'. The first row has 'order_no' 201, 'item_id' NULL, and 'qty' 2. The subsequent rows are 202, 203, 204, and 205, with item IDs 302, 303, 304, and 305 respectively, and quantities 1, 3, 2, and 1. The rows are alternatingly highlighted in white and light blue.

| | order_no | item_id | qty |
|---|----------|---------|-----|
| ▶ | 201 | NULL | 2 |
| | 202 | 302 | 1 |
| | 203 | 303 | 3 |
| | 204 | 304 | 2 |
| | 205 | 305 | 1 |

```
commit;
```

PROGRAM 7. BOOK DEALER DATABASE

The following tables are maintained by a book dealer:

AUTHOR(author-id: int, name: String, city: String, country: String)

PUBLISHER(publisher-id: int, name: String, city: String, country: String)

CATALOG(book-id: int, title: String, author-id: int, publisher-id: int, category-id: int, year: int, price: int)

CATEGORY(category-id: int, description: String)

ORDER-DETAILS(order-no: int, book-id: int, quantity: int)

- i) Create the above tables by properly specifying the primary keys and the foreign keys.
- ii) Enter at least five tuples for each relation.
- iii) Give the details of the authors who have 2 or more books in the catalog and the price of the books in the catalog and the year of publication is after 2000.
- iv) Find the author of the book which has maximum sales.
- v) Demonstrate how you increase the price of books published by a specific publisher by 10%.

```
create database book_dealer;  
use book_dealer;
```

```
create table author (  
  author_id INT,  
  author_name VARCHAR(20),  
  author_city VARCHAR(20),  
  author_country VARCHAR(20),  
  PRIMARY KEY(author_id));  
desc author;
```

| Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content: | | | | | | |
|---|----------------|-------------|------|-----|---------|-------|
| | Field | Type | Null | Key | Default | Extra |
| ▶ | author_id | int | NO | PRI | NULL | |
| | author_name | varchar(20) | YES | | NULL | |
| | author_city | varchar(20) | YES | | NULL | |
| | author_country | varchar(20) | YES | | NULL | |

```
create table publisher (
publisher_id INT,
publisher_name VARCHAR(20),
publisher_city VARCHAR(20),
publisher_country VARCHAR(20),
PRIMARY KEY(publisher_id));
desc publisher;
```

| Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content: | | | | | | |
|---|-------------------|-------------|------|-----|---------|-------|
| | Field | Type | Null | Key | Default | Extra |
| ▶ | publisher_id | int | NO | PRI | NULL | |
| | publisher_name | varchar(20) | YES | | NULL | |
| | publisher_city | varchar(20) | YES | | NULL | |
| | publisher_country | varchar(20) | YES | | NULL | |

```
create table category (
category_id INT,
description VARCHAR(30),
PRIMARY KEY(category_id));
desc category;
```


Result Grid

Filter Rows:

Export:

Wrap Cell Content:

| | Field | Type | Null | Key | Default | Extra |
|---|-------------|-------------|------|-----|---------|-------|
| ▶ | category_id | int | NO | PRI | NULL | |
| | description | varchar(30) | YES | | NULL | |

```

CREATE TABLE catalog(
book_id INT,
book_title VARCHAR(30),
author_id INT,
publisher_id INT,
category_id INT,
year INT,
price INT,
PRIMARY KEY(book_id),
FOREIGN KEY(author_id) REFERENCES author(author_id),
FOREIGN KEY(publisher_id) REFERENCES publisher(publisher_id),
FOREIGN KEY(category_id) REFERENCES category(category_id) );
desc catalog;

```

Result Grid

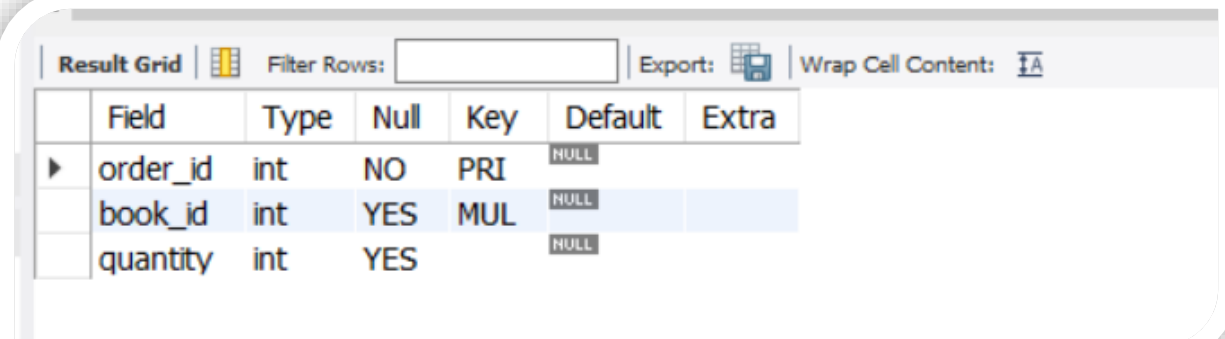
Filter Rows:

Export:

Wrap Cell Content:

| | Field | Type | Null | Key | Default | Extra |
|---|--------------|-------------|------|-----|---------|-------|
| ▶ | book_id | int | NO | PRI | NULL | |
| | book_title | varchar(30) | YES | | NULL | |
| | author_id | int | YES | MUL | NULL | |
| | publisher_id | int | YES | MUL | NULL | |
| | category_id | int | YES | MUL | NULL | |
| | year | int | YES | | NULL | |
| | price | int | YES | | NULL | |

```
CREATE TABLE orderdetails(
order_id INT,
book_id INT,
quantity INT,
PRIMARY KEY(order_id),
FOREIGN KEY(book_id) REFERENCES catalog(book_id));
desc orderdetails;
```



| | Field | Type | Null | Key | Default | Extra |
|---|----------|------|------|-----|---------|-------|
| ▶ | order_id | int | NO | PRI | NULL | |
| | book_id | int | YES | MUL | NULL | |
| | quantity | int | YES | | NULL | |

```
INSERT INTO author (author_id,author_name,author_city,author_country) VALUES
(1001,'JK Rowling','London','England'),
(1002,'Chetan Bhagat','Mumbai','India'),
(1003,'John McCarthy','Chicago','USA'),
(1004,'Dan Brown','California','USA') ;
select * from author;
```

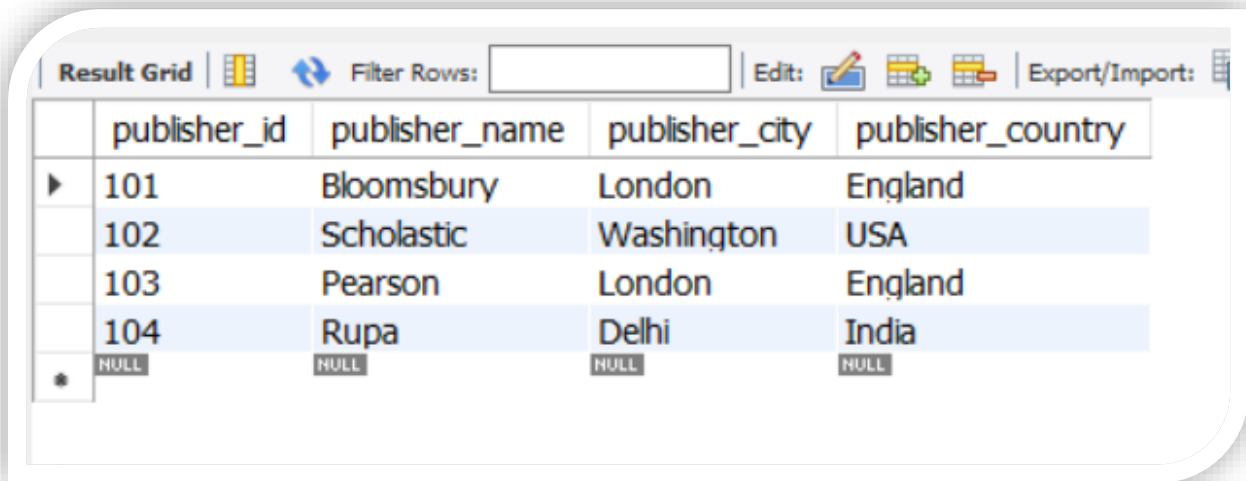


| | author_id | author_name | author_city | author_country |
|---|-----------|---------------|-------------|----------------|
| ▶ | 1001 | JK Rowling | London | England |
| | 1002 | Chetan Bhagat | Mumbai | India |
| | 1003 | John McCarthy | Chicago | USA |
| | 1004 | Dan Brown | California | USA |
| * | NULL | NULL | NULL | NULL |

```

INSERT INTO publisher
(publisher_id,publisher_name,publisher_city,publisher_country) VALUES
(101,'Bloomsbury','London','England'),
(102,'Scholastic','Washington','USA'),
(103,'Pearson','London','England'),
(104,'Rupa','Delhi','India');
select * from publisher;

```



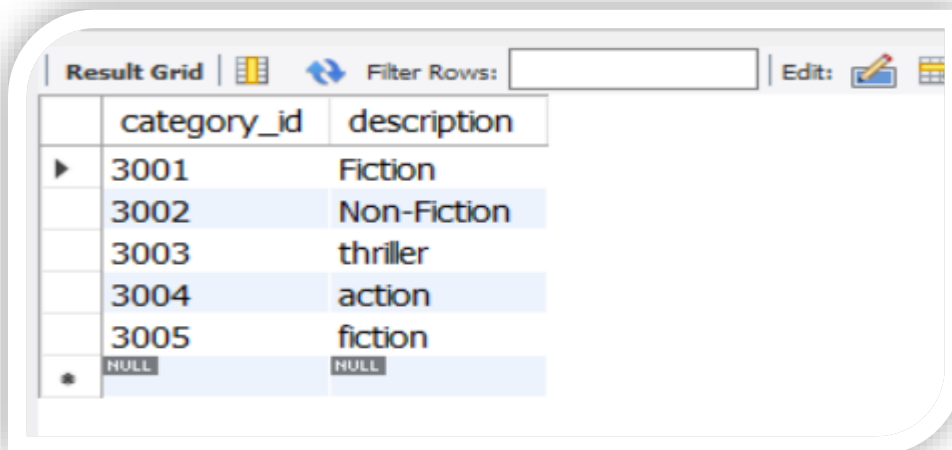
The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the data for the 'publisher' table. The columns are 'publisher_id', 'publisher_name', 'publisher_city', and 'publisher_country'. There are four data rows and one row for NULL values. The interface includes a 'Filter Rows' search bar, an 'Edit' button, and an 'Export/Import' button.

| | publisher_id | publisher_name | publisher_city | publisher_country |
|---|--------------|----------------|----------------|-------------------|
| ▶ | 101 | Bloomsbury | London | England |
| | 102 | Scholastic | Washington | USA |
| | 103 | Pearson | London | England |
| | 104 | Rupa | Delhi | India |
| ★ | NULL | NULL | NULL | NULL |

```

INSERT INTO category (category_id,description) VALUES
(3001,'Fiction'),
(3002,'Non-Fiction'),
(3003,'thriller'),
(3004,'action'),
(3005,'fiction') ;
select * from category;

```

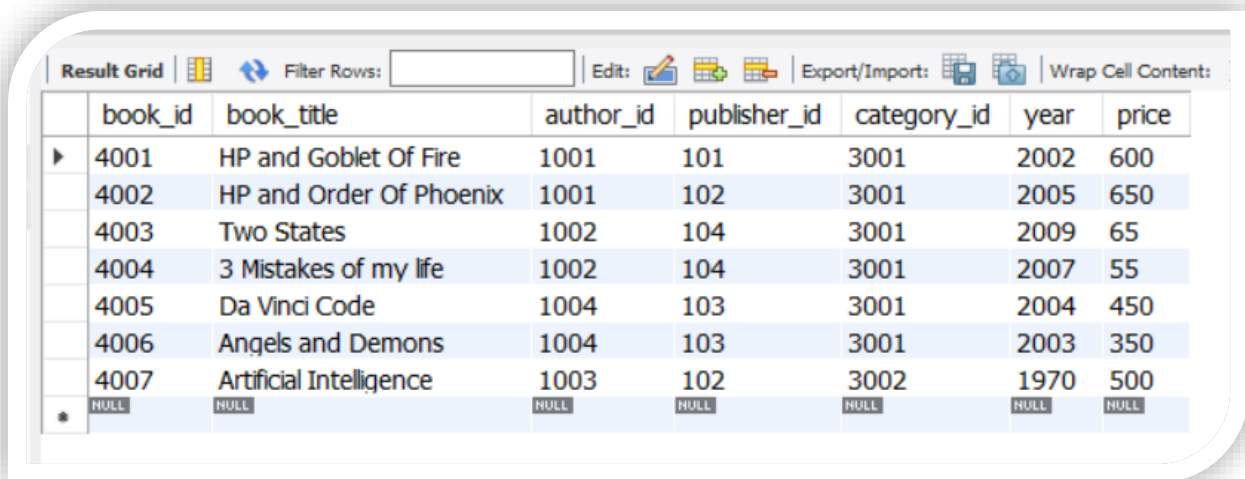


The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the data for the 'category' table. The columns are 'category_id' and 'description'. There are five data rows and one row for NULL values. The interface includes a 'Filter Rows' search bar, an 'Edit' button, and a 'Table' icon.

| | category_id | description |
|---|-------------|-------------|
| ▶ | 3001 | Fiction |
| | 3002 | Non-Fiction |
| | 3003 | thriller |
| | 3004 | action |
| | 3005 | fiction |
| ★ | NULL | NULL |

INSERT INTO catalog

```
(book_id,book_title,author_id,publisher_id,category_id,year,price) VALUES  
(4001,'HP and Goblet Of Fire',1001,101,3001,2002,600),  
(4002,'HP and Order Of Phoenix',1001,102,3001,2005,650),  
(4003,'Two States',1002,104,3001,2009,65),  
(4004,'3 Mistakes of my life',1002,104,3001,2007,55),  
(4005,'Da Vinci Code',1004,103,3001,2004,450),  
(4006,'Angels and Demons',1004,103,3001,2003,350),  
(4007,'Artificial Intelligence',1003,102,3002,1970,500);  
select * from catalog;
```



| | book_id | book_title | author_id | publisher_id | category_id | year | price |
|---|---------|-------------------------|-----------|--------------|-------------|------|-------|
| ▶ | 4001 | HP and Goblet Of Fire | 1001 | 101 | 3001 | 2002 | 600 |
| | 4002 | HP and Order Of Phoenix | 1001 | 102 | 3001 | 2005 | 650 |
| | 4003 | Two States | 1002 | 104 | 3001 | 2009 | 65 |
| | 4004 | 3 Mistakes of my life | 1002 | 104 | 3001 | 2007 | 55 |
| | 4005 | Da Vinci Code | 1004 | 103 | 3001 | 2004 | 450 |
| | 4006 | Angels and Demons | 1004 | 103 | 3001 | 2003 | 350 |
| | 4007 | Artificial Intelligence | 1003 | 102 | 3002 | 1970 | 500 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

INSERT INTO orderdetails (order_id,book_id,quantity) VALUES

```
(5001,4001,5),  
(5002,4002,7),  
(5003,4003,15),  
(5004,4004,11),  
(5005,4005,9),  
(5006,4006,8),  
(5007,4007,2),  
(5008,4004,3) ;
```

select * from orderdetails;

| Result Grid | | | |
|--------------|----------|---------|----------|
| Filter Rows: | | | |
| | order_id | book_id | quantity |
| ▶ | 5001 | 4001 | 5 |
| | 5002 | 4002 | 7 |
| | 5003 | 4003 | 15 |
| | 5004 | 4004 | 11 |
| | 5005 | 4005 | 9 |
| | 5006 | 4006 | 8 |
| | 5007 | 4007 | 2 |
| | 5008 | 4004 | 3 |
| ✱ | NULL | NULL | NULL |

```
select a.author_id,a.author_city,a.author_name,a.author_city
from author a join catalog c on a.author_id=c.author_id where
year>2000 and c.author_id in(select c.author_id from catalog having
count(c.author_id)>2 and price>500 );
```

| Result Grid | | | | |
|------------------------------|-----------|-------------|-------------|-------------|
| Filter Rows: | | | | |
| Export: Wrap Cell Content: | | | | |
| | author_id | author_city | author_name | author_city |
| ▶ | 1001 | London | JK Rowling | London |
| | 1001 | London | JK Rowling | London |

```
select a.author_id from author a join catalog c on
a.author_id=c.author_id join orderdetails o on c.book_id=o.book_id
having max(o.quantity);
```

| | | |
|-------------|--------------|---------|
| Result Grid | Filter Rows: | Export: |
| author_id | | |
| ▶ 1001 | | |

```
update catalog set price=(0.1*price)+price where publisher_id=2001;
select * from catalog;
commit;
```

Result Grid

Filter Rows:

Edit

Export/Import:

Wrap Cell Content:

| | book_id | book_title | author_id | publisher_id | category_id | year | price |
|---|---------|-------------------------|-----------|--------------|-------------|------|-------|
| ▶ | 4001 | HP and Goblet Of Fire | 1001 | 101 | 3001 | 2002 | 600 |
| | 4002 | HP and Order Of Phoenix | 1001 | 102 | 3001 | 2005 | 650 |
| | 4003 | Two States | 1002 | 104 | 3001 | 2009 | 65 |
| | 4004 | 3 Mistakes of my life | 1002 | 104 | 3001 | 2007 | 55 |
| | 4005 | Da Vinci Code | 1004 | 103 | 3001 | 2004 | 450 |
| | 4006 | Angels and Demons | 1004 | 103 | 3001 | 2003 | 350 |
| | 4007 | Artificial Intelligence | 1003 | 102 | 3002 | 1970 | 500 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

PROGRAM 8: STUDENT ENROLLMENT DATABASE

Consider the following database of student enrollment in courses and books adopted for each course.

STUDENT (regno: String, name: String, major: String, bdate: date)

COURSE (course #: int, cname: String, dept: String)

ENROLL (regno: String, cname: String, sem: int, marks: int)

BOOK_ADOPTION (course #: int, sem: int, book-ISBN: int)

TEXT(book-ISBN:int, book-title:String, publisher:String, author:String)

- i) Create the above tables by properly specifying the primary keys and the foreign keys.
- ii) Enter at least five tuples for each relation.
- iii) Demonstrate how you add a new text book to the database and make this book be adopted by some department.
- iv) Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.
- v) List any department that has all its adopted books published by a specific publisher.

```
create database Stud_Enrollment;  
use Stud_Enrollment;
```

```
CREATE TABLE student(  
  regno VARCHAR(30),  
  name VARCHAR(50),  
  major VARCHAR(30),  
  bdate DATE,  
  PRIMARY KEY (regno));  
desc student;
```

| Result Grid Filter Rows: Export: Wrap Cell Content: | | | | | | |
|---|-------|-------------|------|-----|---------|-------|
| | Field | Type | Null | Key | Default | Extra |
| ▶ | regno | varchar(30) | NO | PRI | NULL | |
| | name | varchar(50) | YES | | NULL | |
| | major | varchar(30) | YES | | NULL | |
| | bdate | date | YES | | NULL | |

```
CREATE TABLE course(
  courseno INT,
  cname VARCHAR(20),
  dept VARCHAR(20),
  PRIMARY KEY (courseno));
desc course;
```

| Result Grid Filter Rows: Export: Wrap Cell Content: | | | | | | |
|---|----------|-------------|------|-----|---------|-------|
| | Field | Type | Null | Key | Default | Extra |
| ▶ | courseno | int | NO | PRI | NULL | |
| | cname | varchar(20) | YES | | NULL | |
| | dept | varchar(20) | YES | | NULL | |

```
CREATE TABLE enroll(
  regno VARCHAR(15),
  courseno INT,
  sem INT(3),
  marks INT(4),
  PRIMARY KEY (regno,courseno),
  FOREIGN KEY (regno) REFERENCES student (regno),
  FOREIGN KEY (courseno) REFERENCES course (courseno));
desc enroll;
```

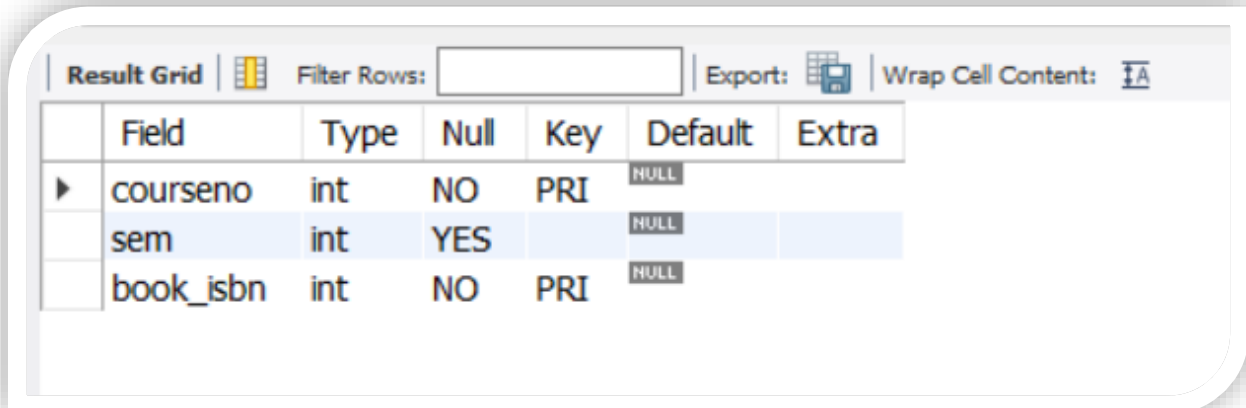

| Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content: | | | | | | |
|---|----------|-------------|------|-----|---------|-------|
| | Field | Type | Null | Key | Default | Extra |
| ▶ | regno | varchar(15) | NO | PRI | NULL | |
| | courseno | int | NO | PRI | NULL | |
| | sem | int | YES | | NULL | |
| | marks | int | YES | | NULL | |

```
CREATE TABLE text(
book_isbn INT(5),
book_title VARCHAR(20),
publisher VARCHAR(20),
author VARCHAR(20),
PRIMARY KEY (book_isbn));
desc text;
```

| Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content: | | | | | | |
|---|------------|-------------|------|-----|---------|-------|
| | Field | Type | Null | Key | Default | Extra |
| ▶ | book_isbn | int | NO | PRI | NULL | |
| | book_title | varchar(20) | YES | | NULL | |
| | publisher | varchar(20) | YES | | NULL | |
| | author | varchar(20) | YES | | NULL | |

```
CREATE TABLE book_adoption(
courseno INT,
sem INT(3),
book_isbn INT(5),
PRIMARY KEY (courseno,book_isbn),
FOREIGN KEY (courseno) REFERENCES course (courseno) on update cascade,
FOREIGN KEY (book_isbn) REFERENCES text(book_isbn)on update cascade
);
```

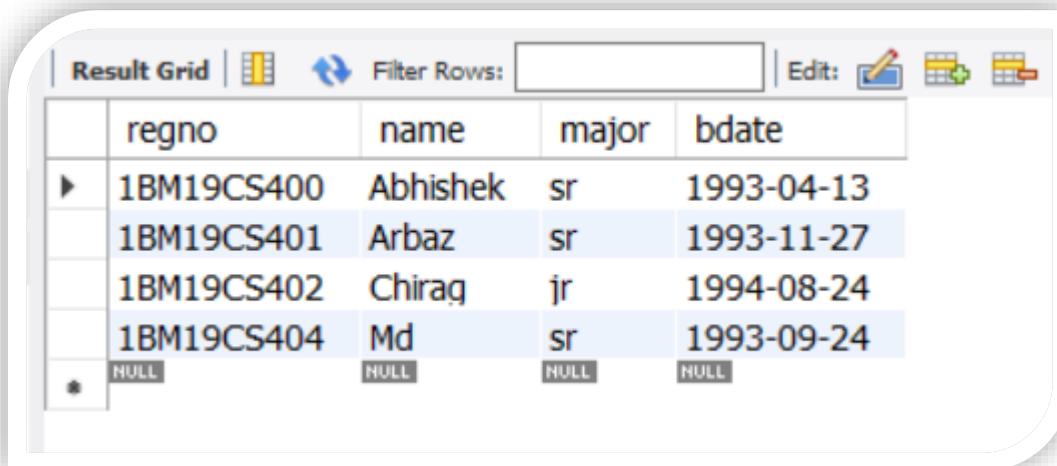
desc book_adoption;



The screenshot shows a 'Result Grid' window with a toolbar at the top containing 'Filter Rows:', 'Export:', and 'Wrap Cell Content:'. The table structure is as follows:

| | Field | Type | Null | Key | Default | Extra |
|---|-----------|------|------|-----|---------|-------|
| ▶ | courseno | int | NO | PRI | NULL | |
| | sem | int | YES | | NULL | |
| | book_isbn | int | NO | PRI | NULL | |

```
INSERT INTO student (regno,name,major,bdate) VALUES
('1BM19CS404','Md','sr','19930924'),
('1BM19CS401','Arbaz','sr','19931127'),
('1BM19CS400','Abhishek','sr','19930413'),
('1BM19CS402','Chirag','jr','19940824');
select * from student;
```



The screenshot shows a 'Result Grid' window with a toolbar at the top containing 'Filter Rows:', 'Edit:', and other icons. The table data is as follows:

| | regno | name | major | bdate |
|---|------------|----------|-------|------------|
| ▶ | 1BM19CS400 | Abhishek | sr | 1993-04-13 |
| | 1BM19CS401 | Arbaz | sr | 1993-11-27 |
| | 1BM19CS402 | Chirag | jr | 1994-08-24 |
| | 1BM19CS404 | Md | sr | 1993-09-24 |
| * | NULL | NULL | NULL | NULL |

```
INSERT INTO course VALUES (111,'OS','CSE'),
(112,'TFCS','CSE'),
(113,'ADA','ISE'),
(114,'DBMS','CSE'),
(115,'C','ECE');
select * from course;
```

| Result Grid | | | |
|--------------|----------|-------|------|
| Filter Rows: | | | |
| | courseno | cname | dept |
| ▶ | 111 | OS | CSE |
| | 112 | TFCS | CSE |
| | 113 | ADA | ISE |
| | 114 | DBMS | CSE |
| | 115 | C | ECE |
| ★ | NULL | NULL | NULL |

```

INSERT INTO text (book_isbn,book_title,publisher,author)VALUES
(10,'DATABASE SYSTEMS','PEARSON','SCHIELD'),
(900,'OPERATING SYS','PEARSON','LELAND'),
(901,'CIRCUITS','HALL INDIA','BOB'),
(902,'SYSTEM SOFTWARE','PETERSON','JACOB'),
(903,'SCHEDULING','PEARSON','PATIL'),
(904,'DATABASE SYSTEMS','PEARSON','JACOB'),
(905,'DATABASE MANAGER','PEARSON','BOB'),
(906,'SIGNALS','HALL INDIA','SUMIT');
select * from text;

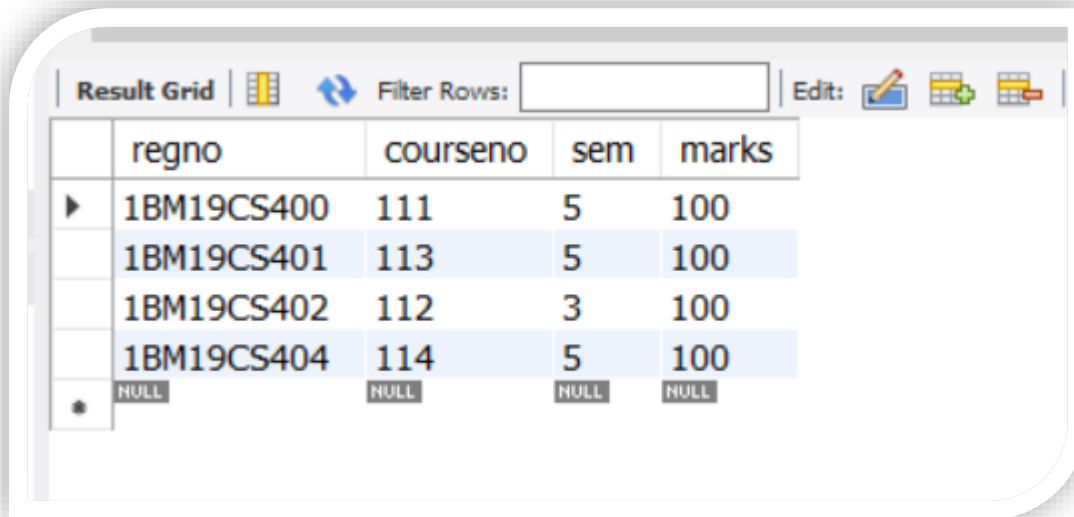
```

| Result Grid | | | | |
|--------------|-----------|------------------|------------|---------|
| Filter Rows: | | | | |
| | book_isbn | book_title | publisher | author |
| ▶ | 10 | DATABASE SYSTEMS | PEARSON | SCHIELD |
| | 900 | OPERATING SYS | PEARSON | LELAND |
| | 901 | CIRCUITS | HALL INDIA | BOB |
| | 902 | SYSTEM SOFTWARE | PETERSON | JACOB |
| | 903 | SCHEDULING | PEARSON | PATIL |
| | 904 | DATABASE SYSTEMS | PEARSON | JACOB |
| | 905 | DATABASE MANAGER | PEARSON | BOB |
| | 906 | SIGNALS | HALL INDIA | SUMIT |
| ★ | NULL | NULL | NULL | NULL |

```

INSERT INTO enroll (regno,courseno,sem,marks) VALUES
('1BM19CS404',114,5,100),
('1BM19CS401',113,5,100),
('1BM19CS400',111,5,100),
('1BM19CS402',112,3,100);
select * from enroll;

```



| | regno | courseno | sem | marks |
|---|------------|----------|------|-------|
| ▶ | 1BM19CS400 | 111 | 5 | 100 |
| | 1BM19CS401 | 113 | 5 | 100 |
| | 1BM19CS402 | 112 | 3 | 100 |
| | 1BM19CS404 | 114 | 5 | 100 |
| * | NULL | NULL | NULL | NULL |

```

INSERT INTO book_adoption (courseno,sem,book_isbn) VALUES
(111,5,900),
(111,5,903),
(111,5,904),
(112,3,901),
(113,3,10),
(114,5,905),
(113,5,902),
(115,3,906);
select * from book_adoption;
commit;

```

| Result Grid | | | |
|--------------|----------|------|-----------|
| Filter Rows: | | | |
| | courseno | sem | book_isbn |
| ▶ | 111 | 5 | 900 |
| | 111 | 5 | 903 |
| | 111 | 5 | 904 |
| | 112 | 3 | 901 |
| | 113 | 3 | 10 |
| | 113 | 5 | 902 |
| | 114 | 5 | 905 |
| | 115 | 3 | 906 |
| • | NULL | NULL | NULL |

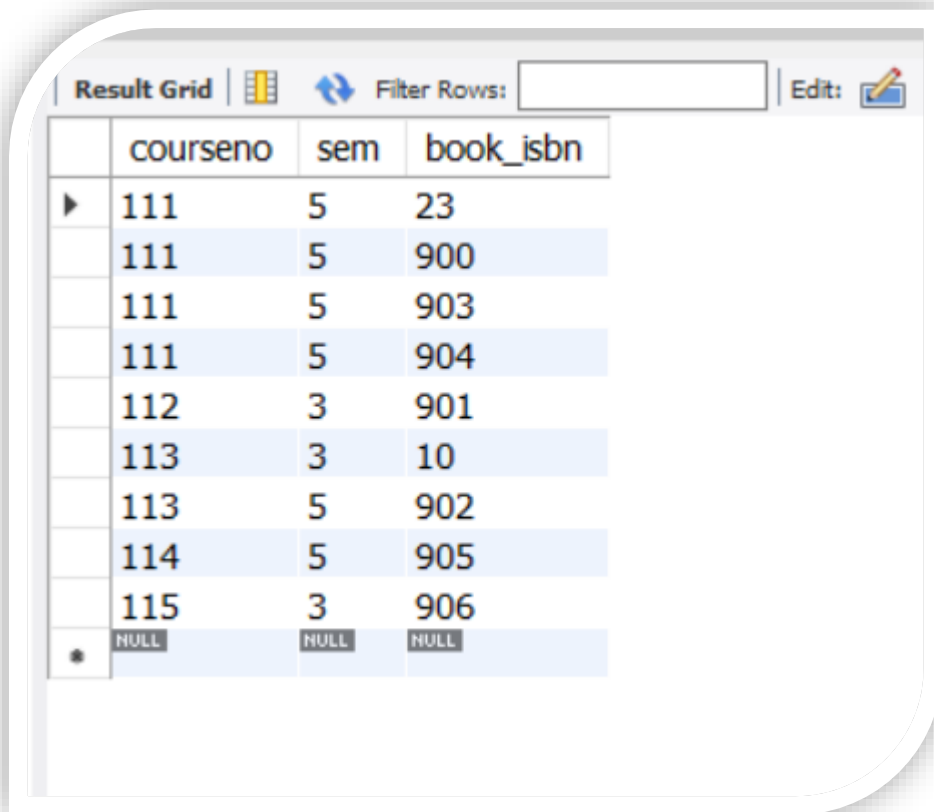
/*QUERIES:

01. Demonstrate how you add a new text book to the database and make this book be adopted by some department */

```
INSERT INTO text (book_isbn, book_title, publisher,author) VALUES
(23,'ADA','PEARSON','SCHIELD');
SELECT * FROM TEXT;
```

| Result Grid | | | | |
|--------------|-----------|------------------|------------|---------|
| Filter Rows: | | | | |
| | book_isbn | book_title | publisher | author |
| ▶ | 10 | DATABASE SYSTEMS | PEARSON | SCHIELD |
| | 23 | ADA | PEARSON | SCHIELD |
| | 900 | OPERATING SYS | PEARSON | LELAND |
| | 901 | CIRCUITS | HALL INDIA | BOB |
| | 902 | SYSTEM SOFTWARE | PETERSON | JACOB |
| | 903 | SCHEDULING | PEARSON | PATIL |
| | 904 | DATABASE SYSTEMS | PEARSON | JACOB |
| | 905 | DATABASE MANAGER | PEARSON | BOB |
| | 906 | SIGNALS | HALL INDIA | SUMIT |
| • | NULL | NULL | NULL | NULL |

```
INSERT INTO book_adoption (courseno,sem,book_isbn) VALUES (111,5,23);
SELECT * FROM BOOK_ADOPTION;
```



| | courseno | sem | book_isbn |
|---|----------|------|-----------|
| ▶ | 111 | 5 | 23 |
| | 111 | 5 | 900 |
| | 111 | 5 | 903 |
| | 111 | 5 | 904 |
| | 112 | 3 | 901 |
| | 113 | 3 | 10 |
| | 113 | 5 | 902 |
| | 114 | 5 | 905 |
| | 115 | 3 | 906 |
| * | NULL | NULL | NULL |

/*02. Produce a list of text books [include course #, Book ISBN, Book title] in the alphabetical order for courses offered by the “CS “ department that use more than two books. STUDENT ENROLLMENT DATABASE*/

```
SELECT c.courseno,t.book_isbn,t.book_title
FROM course c,book_adoption ba,text t
WHERE c.courseno=ba.courseno
AND ba.book_isbn=t.book_isbn
AND c.dept='CSE'
AND 2<( SELECT COUNT(book_isbn)
        FROM book_adoption b
        WHERE c.courseno=b.courseno)
ORDER BY t.book_title;
```

| Result Grid | | | |
|-----------------------------------|----------|-----------|------------------|
| Filter Rows: <input type="text"/> | | | |
| Export: | | | |
| Wrap Cell Content: | | | |
| | courseno | book_isbn | book_title |
| ▶ | 111 | 23 | ADA |
| | 111 | 904 | DATABASE SYSTEMS |
| | 111 | 900 | OPERATING SYS |
| | 111 | 903 | SCHEDULING |

/*03. List any department that has all its adopted books published by a specific publisher.
*/

```
SELECT DISTINCT c.dept FROM course c
WHERE c.dept IN ( SELECT c.dept FROM course
c,book_adoption b,text t WHERE c.courseno=b.courseno
AND t.book_isbn=b.book_isbn
AND t.publisher='PEARSON') AND c.dept
NOT IN (SELECT c.dept FROM course c,book_adoption b,text t
WHERE c.courseno=b.courseno
AND t.book_isbn=b.book_isbn AND t.publisher ='PEARSON');
```

| dept |
|------|
| CSE |
| ISE |
| ECE |

PROGRAM 9: MOVIE DATABASE

Consider the schema for Movie Database:

ACTOR(Act_id, Act_Name, Act_Gender)

DIRECTOR(Dir_id, Dir_Name, Dir_Phone)

MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)

MOVIE_CAST(Act_id, Mov_id, Role)

RATING(Mov_id, Rev_Stars)

Write SQL queries to

1. List the titles of all movies directed by 'Hitchcock'.
2. Find the movie names where one or more actors acted in two or more movies.
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.
5. Update rating of all movies directed by 'Steven Spielberg' to 5.

```
create database m_movie;  
use m_movie;
```

```
create table actor(act_id int,  
act_name varchar(20),  
act_gender varchar(1),  
primary key(act_id));  
desc actor;
```

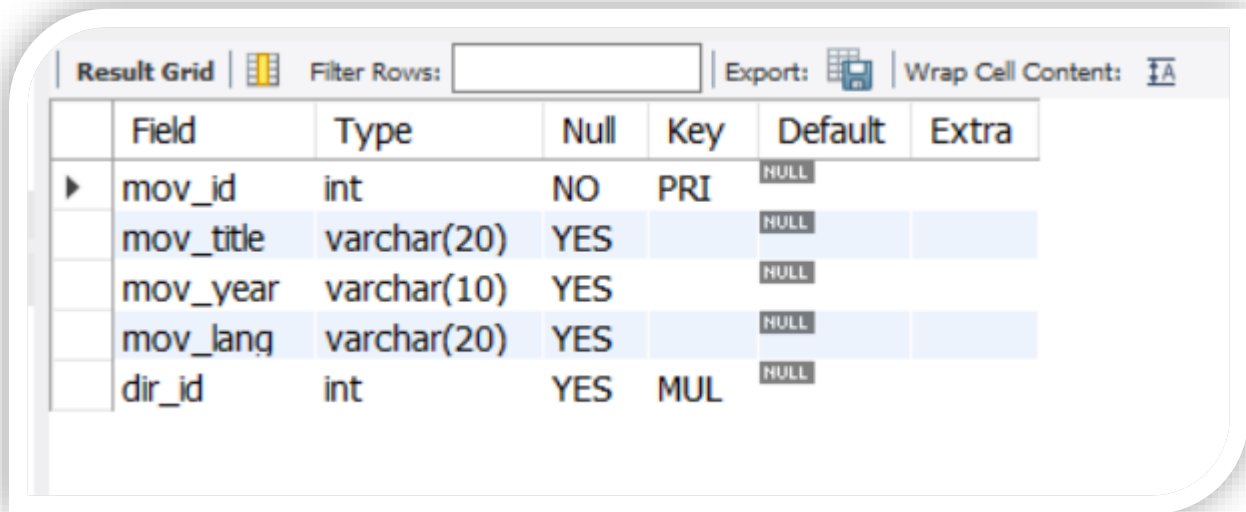

| Result Grid Filter Rows: Export: Wrap Cell Content: | | | | | | |
|---|------------|-------------|------|-----|---------|-------|
| | Field | Type | Null | Key | Default | Extra |
| ▶ | act_id | int | NO | PRI | NULL | |
| | act_name | varchar(20) | YES | | NULL | |
| | act_gender | varchar(1) | YES | | NULL | |

```
create table director(
dir_id int,
dir_name varchar(20),
dir_phone varchar(10),
primary key(dir_id));
desc director;
```

| Result Grid Filter Rows: Export: Wrap Cell Content: | | | | | | |
|---|-----------|-------------|------|-----|---------|-------|
| | Field | Type | Null | Key | Default | Extra |
| ▶ | dir_id | int | NO | PRI | NULL | |
| | dir_name | varchar(20) | YES | | NULL | |
| | dir_phone | varchar(10) | YES | | NULL | |

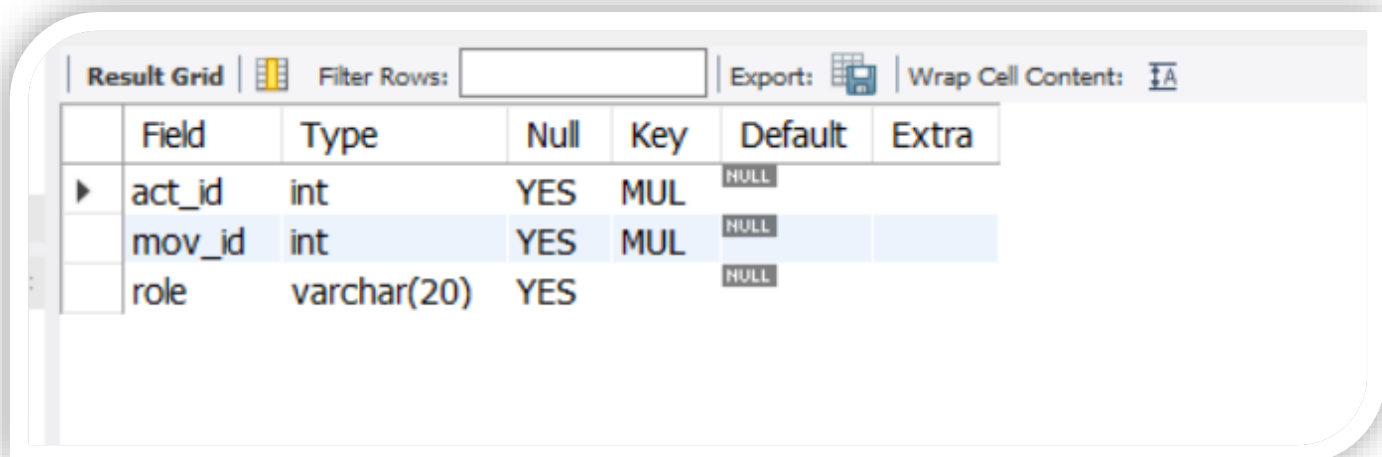
```
create table movies(
mov_id int,
mov_title varchar(20),
mov_year varchar(10),
mov_lang varchar(20),
dir_id int,
primary key(mov_id),
foreign key(dir_id) references director(dir_id));
```

desc movies;



| | Field | Type | Null | Key | Default | Extra |
|---|-----------|-------------|------|-----|---------|-------|
| ▶ | mov_id | int | NO | PRI | NULL | |
| | mov_title | varchar(20) | YES | | NULL | |
| | mov_year | varchar(10) | YES | | NULL | |
| | mov_lang | varchar(20) | YES | | NULL | |
| | dir_id | int | YES | MUL | NULL | |

```
create table movie_cast(  
  act_id int,  
  mov_id int,  
  role varchar(20),  
  foreign key(act_id) references actor(act_id),  
  foreign key(mov_id) references movies(mov_id));  
desc movie_cast;
```



| | Field | Type | Null | Key | Default | Extra |
|---|--------|-------------|------|-----|---------|-------|
| ▶ | act_id | int | YES | MUL | NULL | |
| | mov_id | int | YES | MUL | NULL | |
| | role | varchar(20) | YES | | NULL | |

```

create table rating(
mov_id int,
rev_stars int,
foreign key(mov_id) references movies(mov_id));
desc rating;

```

| Result Grid | | Filter Rows: | | Export: | | Wrap Cell Content: | |
|-------------|-----------|--------------|------|---------|---------|--------------------|--|
| | Field | Type | Null | Key | Default | Extra | |
| ▶ | mov_id | int | YES | MUL | NULL | | |
| | rev_stars | int | YES | | NULL | | |

```

INSERT INTO actor (act_id, act_name, act_gender) VALUES ('101', 'Varun
Dhavan', 'M');
INSERT INTO actor (act_id, act_name, act_gender) VALUES ('102',
'shraddha kapoor', 'F');
INSERT INTO actor (act_id, act_name, act_gender) VALUES ('103',
'Shahid', 'M');
INSERT INTO actor (act_id, act_name, act_gender) VALUES ('104',
'Akshay', 'M');
INSERT INTO actor (act_id, act_name, act_gender) VALUES ('105',
'Deepika', 'F');
select * from actor;




```

| Result Grid | | Filter Rows: | | Edit: | |
|-------------|--------|-----------------|------------|-------|--|
| | act_id | act_name | act_gender | | |
| ▶ | 101 | Varun Dhavan | M | | |
| | 102 | shraddha kapoor | F | | |
| | 103 | Shahid | M | | |
| | 104 | Akshay | M | | |
| | 105 | Deepika | F | | |
| * | NULL | NULL | NULL | | |

```

INSERT INTO director (dir_id, dir_name, dir_phone) VALUES ('11', 'Karan
', '8987565412');
INSERT INTO director (dir_id, dir_name, dir_phone) VALUES ('12',
'HitchCock', '7865254589');
INSERT INTO director (dir_id, dir_name, dir_phone) VALUES ('13',
'Rajamouli ', '9768483512');
INSERT INTO director (dir_id, dir_name, dir_phone) VALUES ('14', 'Remo
', '8987565412');
INSERT INTO director (dir_id, dir_name, dir_phone) VALUES ('15', 'steven
spielberg ', '9868483512');
select * from director;

```

| Result Grid | | | |
|---|--------|------------------|------------|
| Filter Rows: <input type="text"/> | | | |
| Edit:    | | | |
| | dir_id | dir_name | dir_phone |
| ▶ | 11 | Karan | 8987565412 |
| | 12 | HitchCock | 7865254589 |
| | 13 | Rajamouli | 9768483512 |
| | 14 | Remo | 8987565412 |
| | 15 | steven spielberg | 9868483512 |
| ★ | NULL | NULL | NULL |

```

INSERT INTO movies (mov_id, mov_title, mov_year, mov_lang, dir_id)
VALUES ('1001', 'Street Dancers', '2020-01-20', 'English', '14');
INSERT INTO movies (mov_id, mov_title, mov_year, mov_lang, dir_id)
VALUES ('1002', 'Bahubali', '2015-03-31', 'Telagu', '13');
INSERT INTO movies (mov_id, mov_title, mov_year, mov_lang, dir_id)
VALUES ('1003', 'War House', '1999-01-03', 'English', '12');
INSERT INTO movies (mov_id, mov_title, mov_year, mov_lang, dir_id)
VALUES ('1004', 'Akash', '2008-11-04', 'Kannada', '11');
INSERT INTO movies (mov_id, mov_title, mov_year, mov_lang, dir_id)
VALUES ('1005', 'Street Dancers', '2020-01-20', 'Hindi', '15');
select * from movies;

```

Result Grid

Filter Rows:

Edit:

Export/Import:

| | mov_id | mov_title | mov_year | mov_lang | dir_id |
|---|--------|----------------|------------|----------|--------|
| ▶ | 1001 | Street Dancers | 2020-01-20 | English | 14 |
| | 1002 | Bahubali | 2015-03-31 | Telugu | 13 |
| | 1003 | War House | 1999-01-03 | English | 12 |
| | 1004 | Akash | 2008-11-04 | Kannada | 11 |
| | 1005 | Street Dancers | 2020-01-20 | Hindi | 15 |
| • | NULL | NULL | NULL | NULL | NULL |

```

INSERT INTO movie_cast (act_id, mov_id, role) VALUES ('101', '1001',
'Hero');
INSERT INTO movie_cast (act_id, mov_id, role) VALUES ('102', '1002',
'Heroine');
INSERT INTO movie_cast (act_id, mov_id, role) VALUES ('103', '1003',
'Hero');
INSERT INTO movie_cast (act_id, mov_id, role) VALUES ('104', '1004',
'Hero');
INSERT INTO movie_cast (act_id, mov_id, role) VALUES ('101', '1005',
'Hero');
select * from movie_cast;

```

| Result Grid | | | |
|--------------|--------|--------|---------|
| Filter Rows: | | | |
| | act_id | mov_id | role |
| ▶ | 101 | 1001 | Hero |
| | 102 | 1002 | Heroine |
| | 103 | 1003 | Hero |
| | 104 | 1004 | Hero |
| | 101 | 1005 | Hero |

```

INSERT INTO rating (mov_id, rev_stars) VALUES ('1001', '5');
INSERT INTO rating (mov_id, rev_stars) VALUES ('1002', '1');
INSERT INTO rating (mov_id, rev_stars) VALUES ('1003', '4');
INSERT INTO rating (mov_id, rev_stars) VALUES ('1004', '2');
INSERT INTO rating (mov_id, rev_stars) VALUES ('1005', '3');
select * from rating;

```

| Result Grid | | |
|--------------|--------|-----------|
| Filter Rows: | | |
| | mov_id | rev_stars |
| ▶ | 1001 | 5 |
| | 1002 | 1 |
| | 1003 | 4 |
| | 1004 | 2 |
| | 1005 | 3 |

```

-- 1. List the titles of all movies directed by 'Hitchcock'.
SELECT MOV_TITLE FROM MOVIES WHERE DIR_ID IN (SELECT
DIR_ID FROM DIRECTOR WHERE DIR_NAME = 'HITCHCOCK');

```

| | | | |
|-------------|--------------|---------|--------------------|
| Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
| MOV_TITLE | | | |
| ▶ War House | | | |

-- 2. Find the movie names where one or more actors acted in two or more movies.

```
SELECT MOV_TITLE FROM MOVIES M, MOVIE_CAST MV WHERE
M.MOV_ID=MV.MOV_ID AND ACT_ID IN (SELECT ACT_ID FROM
MOVIE_CAST GROUP BY ACT_ID HAVING count(ACT_ID)>=1)
GROUP BY MOV_TITLE HAVING count(mov_title)>1;
```

| | | | |
|------------------|--------------|---------|--------------------|
| Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
| MOV_TITLE | | | |
| ▶ Street Dancers | | | |

-- 3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

```
SELECT ACT_NAME, MOV_TITLE, MOV_YEAR FROM ACTOR A
JOIN MOVIE_CAST C ON A.ACT_ID=C.ACT_ID JOIN MOVIES M ON
C.MOV_ID=M.MOV_ID WHERE M.MOV_YEAR NOT BETWEEN 2000
AND 2015;
```


| Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
|----------------|----------------|------------|--------------------|
| ACT_NAME | MOV_TITLE | MOV_YEAR | |
| ▶ Varun Dhavan | Street Dancers | 2020-01-20 | |
| Shahid | War House | 1999-01-03 | |
| Varun Dhavan | Street Dancers | 2020-01-20 | |




-- 4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.

```
SELECT m.MOV_TITLE, max(r.rev_stars) FROM MOVIES m natural
join RATING r where r.rev_stars>=1 GROUP BY m.MOV_TITLE ORDER
BY m.MOV_TITLE;
```

| Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
|----------------|------------------|---------|--------------------|
| MOV_TITLE | max(r.rev_stars) | | |
| ▶ Akash | 2 | | |
| Bahubali | 1 | | |
| Street Dancers | 5 | | |
| War House | 4 | | |

-- 5. Update rating of all movies directed by 'Steven Spielberg' to 5

```
UPDATE RATING SET REV_STARS=5 WHERE MOV_ID IN (SELECT
MOV_ID FROM MOVIES WHERE DIR_ID IN (SELECT DIR_ID FROM
DIRECTOR WHERE DIR_NAME = 'STEVEN SPIELBERG'));
```

Result Grid   Filter Rows: Export: 

| | mov_id | rev_stars |
|---|--------|-----------|
| ▶ | 1001 | 5 |
| | 1002 | 1 |
| | 1003 | 4 |
| | 1004 | 2 |
| | 1005 | 3 |

commit;

PROGRAM 10:COLLEGE DATABASE

Consider the schema for College Database:

STUDENT(USN, SName, Address, Phone, Gender)

SEMSEC(SSID, Sem, Sec)

CLASS(USN, SSID)

SUBJECT(Subcode, Title, Sem, Credits)

IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

Write SQL queries to

1. List all the student details studying in fourth semester 'C' section.
2. Compute the total number of male and female students in each semester and in each section.
3. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.
4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.
5. Categorize students based on the following criterion:
If FinalIA = 17 to 20 then CAT = 'Outstanding'
If FinalIA = 12 to 16 then CAT = 'Average'
If FinalIA < 12 then CAT = 'Weak'
Give these details only for 8th semester A, B, and C section students.

```
drop database college;  
create database college;  
use college;  
CREATE TABLE STUDENT (  
USN VARCHAR (10) PRIMARY KEY,  
SNAME VARCHAR (25),  
ADDRESS VARCHAR (25),  
PHONE INT ,  
GENDER CHAR (1));
```

```
CREATE TABLE SEMSEC (  
SSID VARCHAR (5) PRIMARY KEY,  
SEM INT ,  
SEC CHAR (1));
```

```
CREATE TABLE CLASS (  
USN VARCHAR (10),  
SSID VARCHAR (5),  
PRIMARY KEY (USN, SSID),  
FOREIGN KEY (USN) REFERENCES STUDENT (USN),  
FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));
```

```
CREATE TABLE SUBJECT (  
SUBCODE VARCHAR (8),  
TITLE VARCHAR (20),  
SEM INT,  
CREDITS INT,  
PRIMARY KEY (SUBCODE));
```

```
CREATE TABLE IAMARKS (  
USN VARCHAR (10),  
SUBCODE VARCHAR (8),  
SSID VARCHAR (5),  
TEST1 INT,  
TEST2 INT,  
TEST3 INT,  
FINALIA INT,  
PRIMARY KEY (USN, SUBCODE, SSID),  
FOREIGN KEY (USN) REFERENCES STUDENT (USN),  
FOREIGN KEY (SUBCODE) REFERENCES SUBJECT (SUBCODE),  
FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));
```

```
INSERT INTO STUDENT VALUES  
( '1RN13CS020','AKSHAY','BELAGAVI', 1232654578,'M');  
INSERT INTO STUDENT VALUES
```

```

('1RN13CS062','SANDHYA','BENGALURU', 1232654578,'F');
INSERT INTO STUDENT VALUES
('1RN13CS091','TEESHA','BENGALURU', 1232654578,'F');
INSERT INTO STUDENT VALUES
('1RN13CS066','SUPRIYA','MANGALURU', 1232654578,'F');
INSERT INTO STUDENT VALUES
('1RN14CS010','ABHAY','BENGALURU', 1232654578,'M');
INSERT INTO STUDENT VALUES
('1RN14CS032','BHASKAR','BENGALURU', 1232654578,'M');
INSERT INTO STUDENT VALUES
('1RN14CS025','ASMI','BENGALURU', 1232654578,'F');
INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR',
1232654578,'M');
INSERT INTO STUDENT VALUES
('1RN15CS029','CHITRA','DAVANGERE', 1232654578,'F');
INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY',
1232654578,'M');
INSERT INTO STUDENT VALUES
('1RN15CS091','SANTOSH','MANGALURU', 1232654578,'M');
INSERT INTO STUDENT VALUES
('1RN16CS045','ISMAIL','KALBURGI', 1232654578,'M');
INSERT INTO STUDENT VALUES
('1RN16CS088','SAMEERA','SHIMOGA', 1232654578,'F');
INSERT INTO STUDENT VALUES
('1RN16CS122','VINAYAKA','CHIKAMAGALUR', 1232654578,'M');

```

```

INSERT INTO SEMSEC VALUES ('CSE8A', 8,'A');
INSERT INTO SEMSEC VALUES ('CSE8B', 8,'B');
INSERT INTO SEMSEC VALUES ('CSE8C', 8,'C');
INSERT INTO SEMSEC VALUES ('CSE7A', 7,'A');
INSERT INTO SEMSEC VALUES ('CSE7B', 7,'B');
INSERT INTO SEMSEC VALUES ('CSE7C', 7,'C');
INSERT INTO SEMSEC VALUES ('CSE6A', 6,'A');
INSERT INTO SEMSEC VALUES ('CSE6B', 6,'B');
INSERT INTO SEMSEC VALUES ('CSE6C', 6,'C');

```

```
INSERT INTO SEMSEC VALUES ('CSE5A', 5,'A');
INSERT INTO SEMSEC VALUES ('CSE5B', 5,'B');
INSERT INTO SEMSEC VALUES ('CSE5C', 5,'C');
INSERT INTO SEMSEC VALUES ('CSE4A', 4,'A');
INSERT INTO SEMSEC VALUES ('CSE4B', 4,'B');
INSERT INTO SEMSEC VALUES ('CSE4C', 4,'C');
INSERT INTO SEMSEC VALUES ('CSE3A', 3,'A');
INSERT INTO SEMSEC VALUES ('CSE3B', 3,'B');
INSERT INTO SEMSEC VALUES ('CSE3C', 3,'C');
INSERT INTO SEMSEC VALUES ('CSE2A', 2,'A');
INSERT INTO SEMSEC VALUES ('CSE2B', 2,'B');
INSERT INTO SEMSEC VALUES ('CSE2C', 2,'C');
INSERT INTO SEMSEC VALUES ('CSE1A', 1,'A');
INSERT INTO SEMSEC VALUES ('CSE1B', 1,'B');
INSERT INTO SEMSEC VALUES ('CSE1C', 1,'C');
```

```
INSERT INTO CLASS VALUES ('1RN13CS020','CSE8A');
INSERT INTO CLASS VALUES ('1RN13CS062','CSE8A');
INSERT INTO CLASS VALUES ('1RN13CS066','CSE8B');
INSERT INTO CLASS VALUES ('1RN13CS091','CSE8C');
INSERT INTO CLASS VALUES ('1RN14CS010','CSE7A');
INSERT INTO CLASS VALUES ('1RN14CS025','CSE7A');
INSERT INTO CLASS VALUES ('1RN14CS032','CSE7A');
INSERT INTO CLASS VALUES ('1RN15CS011','CSE4A');
INSERT INTO CLASS VALUES ('1RN15CS029','CSE4A');
```

```
INSERT INTO SUBJECT VALUES ('10CS81','ACA', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS82','SSM', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS83','NM', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS84','CC', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS85','PW', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS71','OOAD', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS72','ECS', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS73','PTW', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS74','DWDM', 7, 4);
```

```

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2,
TEST3) VALUES ('1RN13CS091','10CS81','CSE8C', 15, 16, 18);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2,
TEST3) VALUES ('1RN13CS091','10CS82','CSE8C', 12, 19, 14);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2,
TEST3) VALUES ('1RN13CS091','10CS83','CSE8C', 19, 15, 20);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2,
TEST3) VALUES ('1RN13CS091','10CS84','CSE8C', 20, 16, 19);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2,
TEST3) VALUES ('1RN13CS091','10CS85','CSE8C', 15, 15, 12);

```

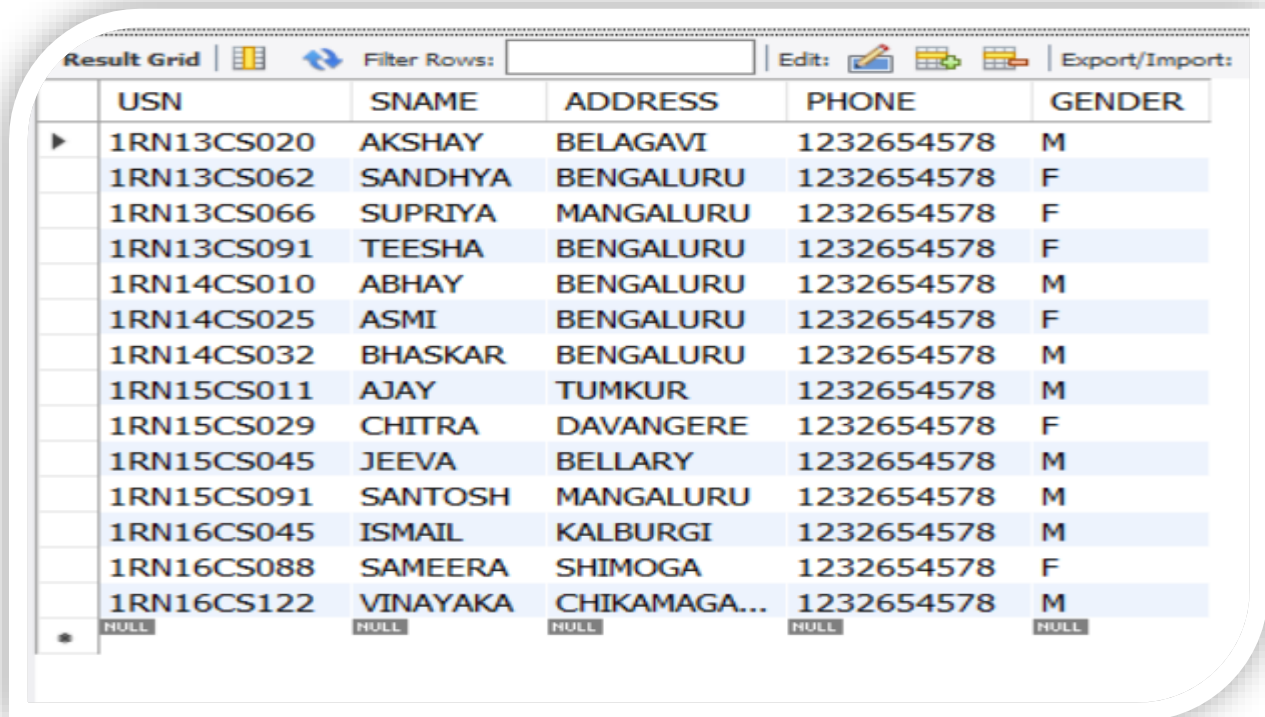
```
select * from iamarks;
```

| Result Grid | | | | | | | |
|--------------|------------|---------|-------|----------------|-------|--------------------|---------|
| Filter Rows: | | Edit: | | Export/Import: | | Wrap Cell Content: | |
| | USN | SUBCODE | SSID | TEST1 | TEST2 | TEST3 | FINALIA |
| ▶ | 1RN13CS091 | 10CS81 | CSE8C | 15 | 16 | 18 | NULL |
| | 1RN13CS091 | 10CS82 | CSE8C | 12 | 19 | 14 | NULL |
| | 1RN13CS091 | 10CS83 | CSE8C | 19 | 15 | 20 | NULL |
| | 1RN13CS091 | 10CS84 | CSE8C | 20 | 16 | 19 | NULL |
| | 1RN13CS091 | 10CS85 | CSE8C | 15 | 15 | 12 | NULL |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

```
select * from subject;
```

| Result Grid | | | | |
|--------------|---------|-------|------|---------|
| Filter Rows: | | Edit: | | |
| | SUBCODE | TITLE | SEM | CREDITS |
| ▶ | 10CS71 | OOAD | 7 | 4 |
| | 10CS72 | ECS | 7 | 4 |
| | 10CS73 | PTW | 7 | 4 |
| | 10CS74 | DWDM | 7 | 4 |
| | 10CS81 | ACA | 8 | 4 |
| | 10CS82 | SSM | 8 | 4 |
| | 10CS83 | NM | 8 | 4 |
| | 10CS84 | CC | 8 | 4 |
| | 10CS85 | PW | 8 | 4 |
| | NULL | NULL | NULL | NULL |

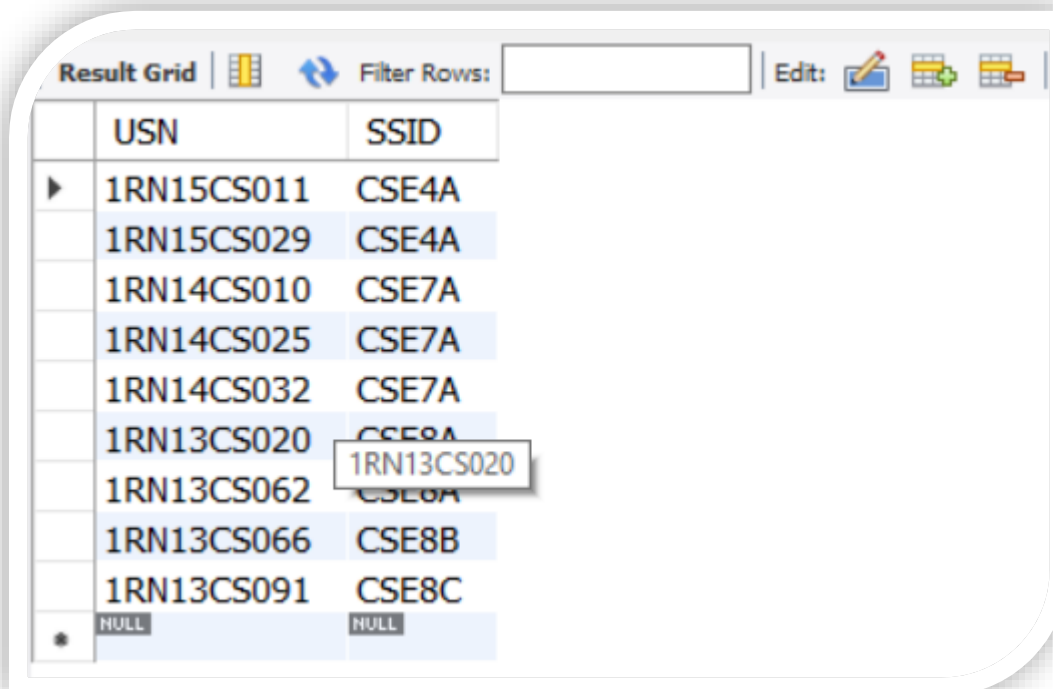
select * from student;



The screenshot shows a database query result grid with the following columns: USN, SNAME, ADDRESS, PHONE, and GENDER. The grid contains 15 rows of student data. The first row is highlighted with a blue arrow pointing to it. The last row shows NULL values for all columns.

| USN | SNAME | ADDRESS | PHONE | GENDER |
|------------|----------|--------------|------------|--------|
| 1RN13CS020 | AKSHAY | BELAGAVI | 1232654578 | M |
| 1RN13CS062 | SANDHYA | BENGALURU | 1232654578 | F |
| 1RN13CS066 | SUPRIYA | MANGALURU | 1232654578 | F |
| 1RN13CS091 | TEESHA | BENGALURU | 1232654578 | F |
| 1RN14CS010 | ABHAY | BENGALURU | 1232654578 | M |
| 1RN14CS025 | ASMI | BENGALURU | 1232654578 | F |
| 1RN14CS032 | BHASKAR | BENGALURU | 1232654578 | M |
| 1RN15CS011 | AJAY | TUMKUR | 1232654578 | M |
| 1RN15CS029 | CHITRA | DAVANGERE | 1232654578 | F |
| 1RN15CS045 | JEEVA | BELLARY | 1232654578 | M |
| 1RN15CS091 | SANTOSH | MANGALURU | 1232654578 | M |
| 1RN16CS045 | ISMAIL | KALBURGI | 1232654578 | M |
| 1RN16CS088 | SAMEERA | SHIMOGA | 1232654578 | F |
| 1RN16CS122 | VINAYAKA | CHIKAMAGA... | 1232654578 | M |
| NULL | NULL | NULL | NULL | NULL |



select * from class;



The screenshot shows a database query result grid with the following columns: USN and SSID. The grid contains 10 rows of class data. The first row is highlighted with a blue arrow pointing to it. The last row shows NULL values for both columns.

| USN | SSID |
|------------|-------|
| 1RN15CS011 | CSE4A |
| 1RN15CS029 | CSE4A |
| 1RN14CS010 | CSE7A |
| 1RN14CS025 | CSE7A |
| 1RN14CS032 | CSE7A |
| 1RN13CS020 | CSE8A |
| 1RN13CS062 | CSE8A |
| 1RN13CS066 | CSE8B |
| 1RN13CS091 | CSE8C |
| NULL | NULL |

select * from semsec;

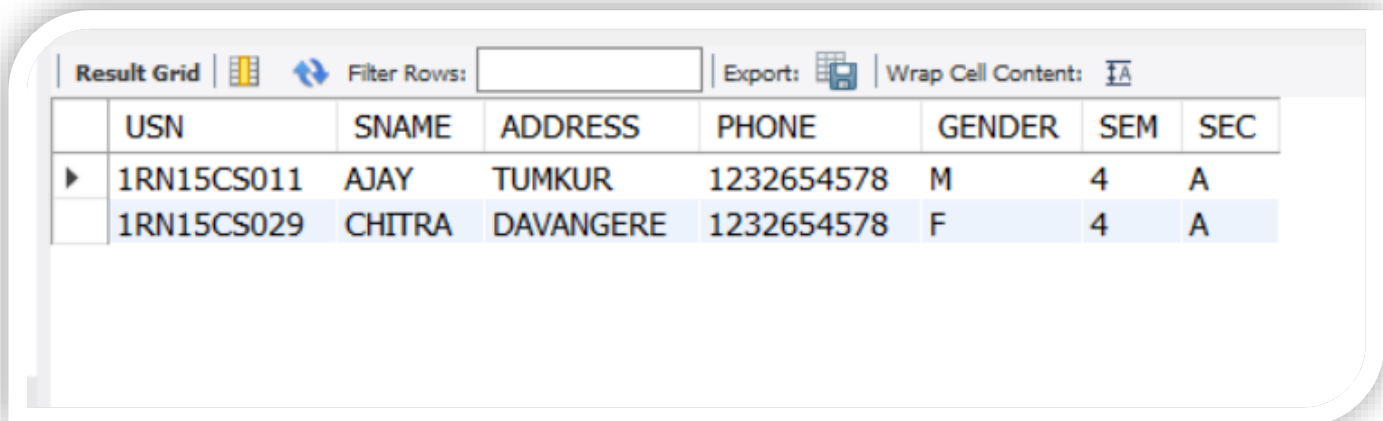
Result Grid   Filter Rows: Edit:

| | SSID | SEM | SEC |
|---|-------|-----|-----|
| ▶ | CSE1A | 1 | A |
| | CSE1B | 1 | B |
| | CSE1C | 1 | C |
| | CSE2A | 2 | A |
| | CSE2B | 2 | B |
| | CSE2C | 2 | C |
| | CSE3A | 3 | A |
| | CSE3B | 3 | B |
| | CSE3C | 3 | C |
| | CSE4A | 4 | A |
| | CSE4B | 4 | B |
| | CSE4C | 4 | C |
| | CSE5A | 5 | A |
| | CSE5B | 5 | B |
| | CSE5C | 5 | C |
| | CSE6A | 6 | A |
| | CSE6B | 6 | B |
| | CSE6C | 6 | C |
| | CSE7A | 7 | A |
| | CSE7B | 7 | B |
| | CSE7C | 7 | C |
| | CSE8A | 8 | A |
| | CSE8B | 8 | B |

| | | | |
|---|-------|------|------|
| | CSE8C | 8 | C |
| • | NULL | NULL | NULL |

/* List all the student details studying in fourth semester 'A' section */

```
SELECT S.*, SS.SEM, SS.SEC
FROM STUDENT S, SEMSEC SS, CLASS C
WHERE S.USN = C.USN AND SS.SSID = C.SSID AND SS.SEM = 4 AND
SS.SEC='A';
```






The screenshot shows a database query result grid with the following data:

| | USN | SNAME | ADDRESS | PHONE | GENDER | SEM | SEC |
|---|------------|--------|-----------|------------|--------|-----|-----|
| ▶ | 1RN15CS011 | AJAY | TUMKUR | 1232654578 | M | 4 | A |
| | 1RN15CS029 | CHITRA | DAVANGERE | 1232654578 | F | 4 | A |




/* Compute the total number of male and female students in each semester and in each section */

```
SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT
FROM STUDENT S, SEMSEC SS, CLASS C
WHERE S.USN = C.USN AND
SS.SSID = C.SSID
GROUP BY SS.SEM, SS.SEC, S.GENDER
ORDER BY SEM;
```

| Result Grid   Filter Rows: <input type="text"/> Export:  | | | | |
|---|-----|-----|--------|-------|
| | SEM | SEC | GENDER | COUNT |
| ▶ | 4 | A | F | 1 |
| | 4 | A | M | 1 |
| | 7 | A | F | 1 |
| | 7 | A | M | 2 |
| | 8 | A | F | 1 |
| | 8 | A | M | 1 |
| | 8 | B | F | 1 |
| | 8 | C | F | 1 |

/* Create a view of Test1 marks of student USN '1BI15CS101' in all subjects */

```
CREATE VIEW STU_TEST1_MARKS_VIEW
AS
SELECT TEST1, SUBCODE
FROM IAMARKS
WHERE USN = '1RN13CS091';
select * from STU_TEST1_MARKS_VIEW;
```

| Result Grid   Filter Rows: <input type="text"/> Export:  | | |
|---|-------|---------|
| | TEST1 | SUBCODE |
| ▶ | 15 | 10CS81 |
| | 12 | 10CS82 |
| | 19 | 10CS83 |
| | 20 | 10CS84 |
| | 15 | 10CS85 |

```

/* Categorize students based on the following criterion:
If FinalIA = 17 to 20 then CAT = 'Outstanding'
If FinalIA = 12 to 16 then CAT = 'Average'
If FinalIA < 12 then CAT = 'Weak'
Give these details only for 8th semester A, B, and C section students */

```

```




update iemarks set finalia=(test1+test2+test2)/3;

```

```

SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,
(CASE
WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING'
WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE'
ELSE 'WEAK'
END) AS CAT
FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB
WHERE S.USN = IA.USN AND
SS.SSID = IA.SSID AND
SUB.SUBCODE = IA.SUBCODE AND
SUB.SEM = 8;

```

| Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content:  | | | | | | |
|--|------------|--------|-----------|------------|--------|-------------|
| | USN | SNAME | ADDRESS | PHONE | GENDER | CAT |
| ▶ | 1RN13CS091 | TEESHA | BENGALURU | 1232654578 | F | AVERAGE |
| | 1RN13CS091 | TEESHA | BENGALURU | 1232654578 | F | OUTSTANDING |
| | 1RN13CS091 | TEESHA | BENGALURU | 1232654578 | F | AVERAGE |
| | 1RN13CS091 | TEESHA | BENGALURU | 1232654578 | F | OUTSTANDING |
| | 1RN13CS091 | TEESHA | BENGALURU | 1232654578 | F | AVERAGE |

Commit