

# **SQL PROJECT**

## **Railway Management System**

### **❖ Abstract :-**

The Railway reservation system facilities the passenger to enquired about the trains available on the basis of source and destination, booking and cancellation of tickets, etc.

The aim of the case study is to design and developed the database maintaining the records of different trains and passengers.

This project contains introduction to railway reservation system.it is the computerized system of reserving the seat of train seat in advance.it is the mainly used for long route online reservation has made the process for the reservation of seat very much easier than ever before.



### **❖ Aim of Project –**

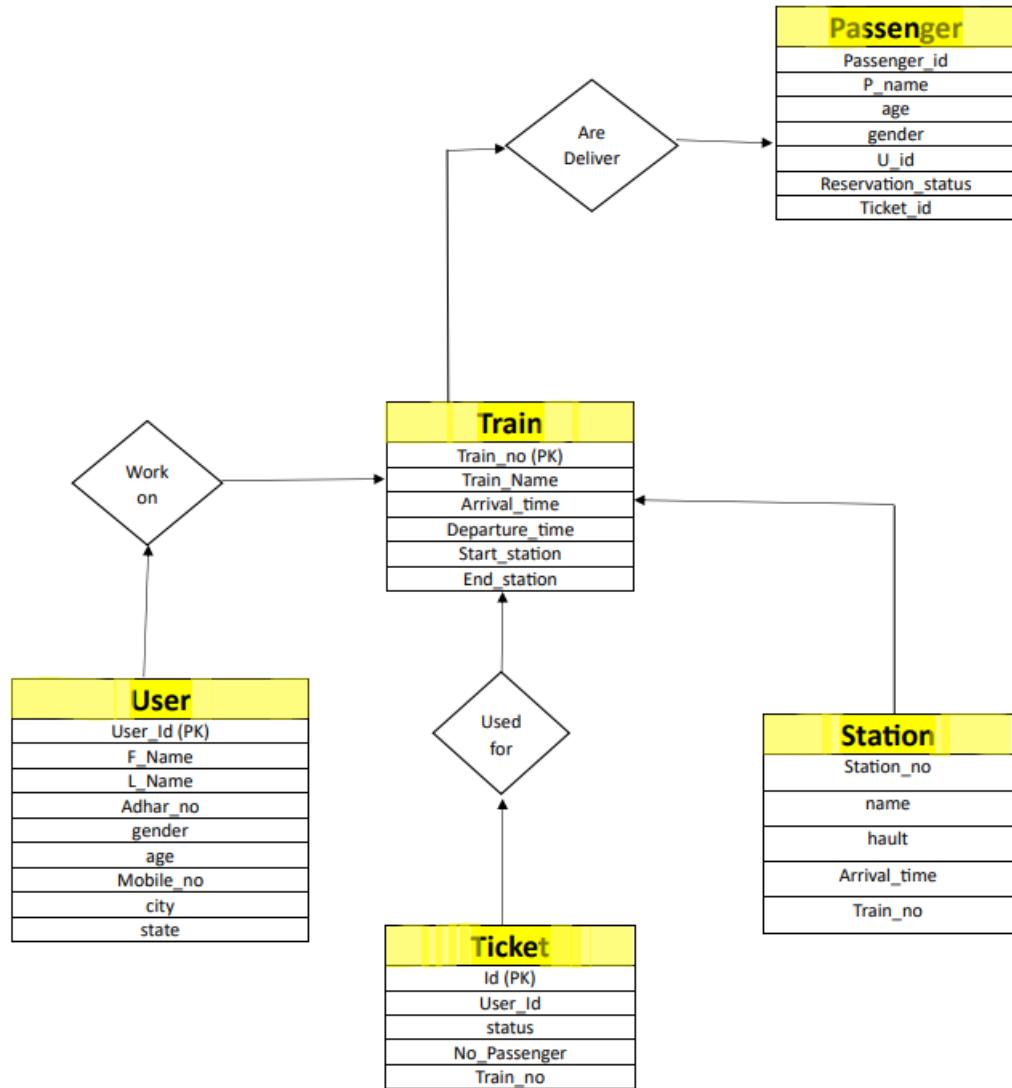
The main aim of this project is to develop a database maintaining the records of different trains and passengers, etc.

This project contains computerized system of reserving the seat of train seat in advanced.

### **❖ Introduction-**

The main purpose of maintain database for railway reservation system is to reduce the manual errors involved in the booking and cancelling of tickets and make it convenient for the customer and providers to maintain the data about their customers and also about the seat available at them. due to automation many loopholes that exist in the manual maintenance of the record can be removed. The speed of obtaining and processing data will be fast. for future expansion the purposed system can be web enabled so that clients can make various enquiries about train between station due to this, sometimes a lot of problems occurs and they are facing many disputes with customers. To solve the above problem, we design database which includes customer details, availability of seat in trains, number of trains and their details.

## ER Diagram:



# **STRUCTURE OF TABLES**

## **Train table -**

The screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. The code pane displays the creation of the 'train' table:

```
25
26 -- create table train
27 • 27 create table train(id int,
28   train_no int primary key,
29   train_name varchar(20),
30   arrival_time time,
31   departure_time time,
32   Start_Station varchar(30),
33   End_Station varchar(30));
34 • desc train;
35
```

The results grid shows the table structure:

Field	Type	Null	Key	Default	Extra
<b>id</b>	int(11)	YES		NULL	
<b>train_no</b>	int(11)	NO	PRI	NULL	
<b>train_name</b>	varchar(20)	YES		NULL	
<b>arrival_time</b>	time	YES		NULL	
<b>departure_time</b>	time	YES		NULL	
<b>Start_Station</b>	varchar(30)	YES		NULL	
<b>End_Station</b>	varchar(30)	YES		NULL	

## **User table –**

The screenshot shows the MySQL Workbench interface with the 'SQL' tab selected. The code pane displays the creation of the 'user' table:

```
5
6 -- create table user
7 create table user(user_id int primary key,
8   first_name varchar(20),
9   last_name varchar(20),
10  address_no varchar(20),
11  gender varchar(1),
12  age int,
13  mobile_no varchar(10),
14  city varchar(20),
15  state varchar(20));
16
```

The results grid shows the table structure:

Field	Type	Null	Key	Default	Extra
<b>user_id</b>	int(11)	NO	PRI	NULL	
<b>first_name</b>	varchar(20)	YES		NULL	
<b>last_name</b>	varchar(20)	YES		NULL	
<b>address_no</b>	varchar(20)	YES		NULL	
<b>gender</b>	varchar(1)	YES		NULL	
<b>age</b>	int(11)	YES		NULL	
<b>mobile_no</b>	varchar(10)	YES		NULL	

## Passengers Table –

The screenshot shows the MySQL Workbench interface with the SQL editor tab selected. The code pane contains the following SQL script:

```
97
98 -- create table passenger
99 •  create table passenger(passenger_id int primary key,
100   p_name varchar(20),
101   age int,
102   gender varchar(5),
103   user_id int,
104   reservatin_status varchar(10),
105   ticket_id int);
106 •  desc passenger;
107 •  insert into passenger values('1','John Doe',20,'M',1,'P','1');
```

The results grid shows the structure of the table:

Field	Type	Null	Key	Default	Extra
passenger_id	int(11)	NO	PRI	NULL	
p_name	varchar(20)	YES			
age	int(11)	YES			
gender	varchar(5)	YES			
user_id	int(11)	YES			
reservatin_status	varchar(10)	YES			
ticket_id	int(11)	YES			

## Station table –

The screenshot shows the MySQL Workbench interface with the SQL editor tab selected. The code pane contains the following SQL script:

```
43 •  drop table train;
44 •  select * from train;
45
46 -- create table station
47 •  create table station(No int,
48   name varchar(20),
49   hault int,
50   arrival_time time,
51   train_no int);
52 •  desc station;
```

The results grid shows the structure of the table:

Field	Type	Null	Key	Default	Extra
No	No	YES			
name	varchar(20)	YES			
hault	int(11)	YES			
arrival_time	time	YES			
train_no	int(11)	YES			

## Ticket table –

The screenshot shows the MySQL Workbench interface with the SQL tab selected. A new table named 'ticket' is being created with the following schema:

```
create table ticket(id int primary key,
user_id int,
status varchar(20),
no_of_passenger int,
train_no int,
ticket_price int);
desc ticket;
```

Two rows of data are inserted into the 'ticket' table:

```
insert into ticket values(1,102,'C',1,12267,15),
(2,104,'NC',1,22204,20),
```

The resulting table structure is displayed in the Result Grid:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
user_id	int(11)	YES		NULL	
status	varchar(20)	YES		NULL	
no_of_passenger	int(11)	YES		NULL	
train_no	int(11)	YES		NULL	
ticket_price	int(11)	YES		NULL	

## Contents of Tables –

### train table -

The screenshot shows the MySQL Workbench interface with the SQL tab selected. A new table named 'train' is being created with the following schema:

```
create table train(id int primary key,
train_no int,
train_name varchar(50),
arrival_time time,
departure_time time,
start_station varchar(50),
end_station varchar(50));
```

Multiple rows of data are inserted into the 'train' table:

```
insert into train values(101,12267,'duronto express','08:35','16:35','Mumbai','Delhi'),
(102,22204,'vande bharat express','06:35','20:15','Mumbai','Pune'),
(103,12268,'mahalaxmi express','06:00','23:40','Kolhapur','Mumbai'),
(104,12953,'deccan express','10:55','17:40','Pune','Mumbai'),
(105,12951,'sahyadri express','08:35','16:35','Kolhapur','Mumbai');
```

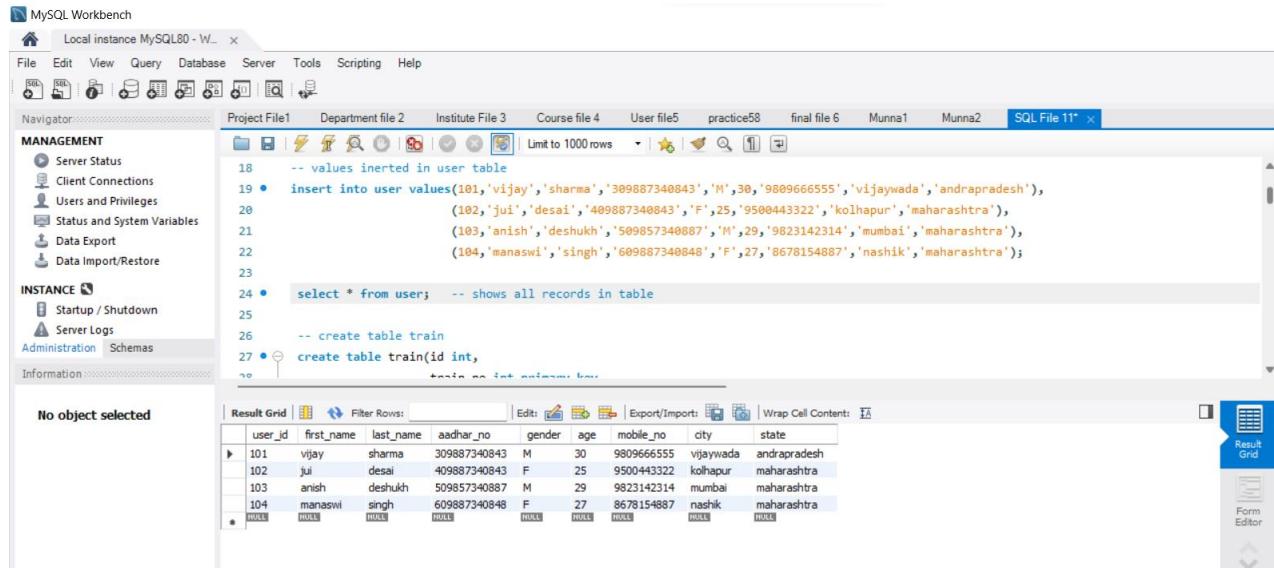
The table is then modified:

```
insert into train(id,train_no,arrival_time,departure_time,start_station,end_station) values(106,12457,'04:15:00','08:30:00','mumbai','chenn');
drop table train;
select * from train;
```

The resulting table structure is displayed in the Result Grid:

id	train_no	train_name	arrival_time	departure_time	Start_Station	End_Station
107	12225	HULL	09:30:00	17:00:00	mumbai	ratnagiri
101	12267	duronto express	08:35:00	16:35:00	Mumbai	Delhi
103	12268	mahalaxmi express	06:00:00	23:40:00	Kolhapur	Mumbai
106	12457	HULL	04:15:00	08:30:00	mumbai	chennai
105	12951	sahyadri express	08:35:00	16:35:00	Kolhapur	Mumbai
104	12953	deccan express	10:55:00	17:40:00	Pune	Mumbai
102	22204	vande bharat express	06:35:00	20:15:00	Mumbai	Pune

## User table –



MySQL Workbench - Local instance MySQL80 - W...

File Edit View Query Database Server Tools Scripting Help

Navigator: Project File1 Department file 2 Institute File 3 Course file 4 User file5 practice58 final file 6 Munna1 Munna2 SQL File 11\* x

**MANAGEMENT**

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

**INSTANCE**

- Startup / Shutdown
- Server Logs
- Administration Schemas

No object selected

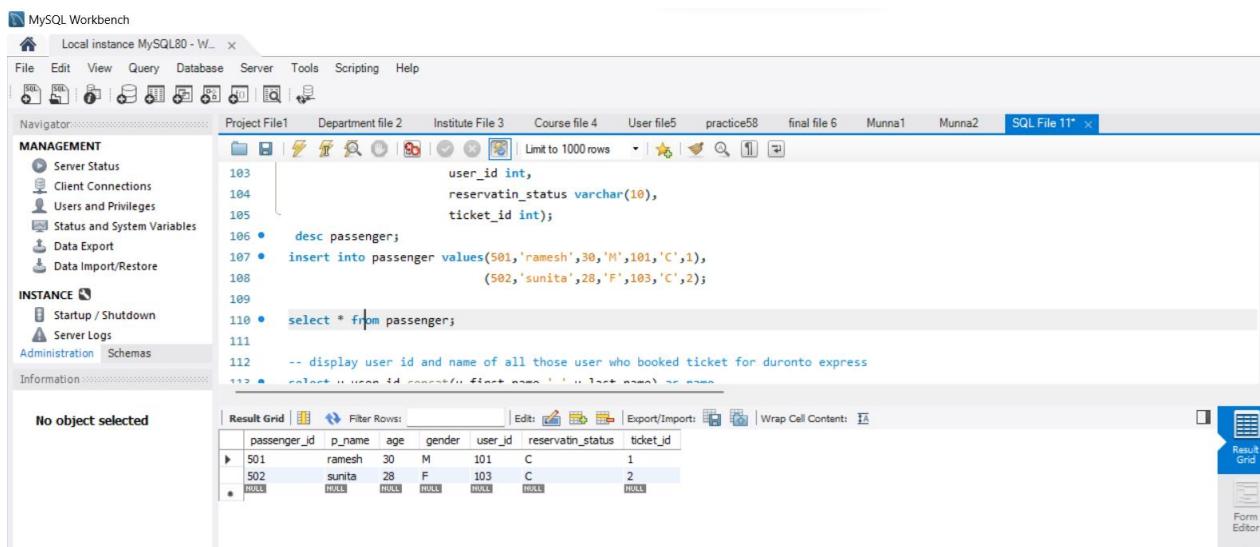
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

user_id	first_name	last_name	aadhar_no	gender	age	mobile_no	city	state
101	vijay	sharma	309887340843	M	30	9809666555	vijaywada	andrapradesh
102	jui	desai	409887340843	F	25	9500443322	kolhapur	maharashtra
103	anish	deshukh	509857340887	M	29	9823142314	mumbai	maharashtra
104	manaswi	singh	609887340848	F	27	8678154887	nashik	maharashtra
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid Form Editor

```
18 -- values inserted in user table
19 • insert into user values(101,'vijay','sharma','309887340843','M',30,'9809666555','vijaywada','andrapradesh'),
20          (102,'jui','desai','409887340843','F',25,'9500443322','kolhapur','maharashtra'),
21          (103,'anish','deshukh','509857340887','M',29,'9823142314','mumbai','maharashtra'),
22          (104,'manaswi','singh','609887340848','F',27,'8678154887','nashik','maharashtra');
23
24 • select * from user; -- shows all records in table
25
26 -- create table train
27 • create table train(id int,
28          train_no int primary key)
```

## Passengers table –



MySQL Workbench - Local instance MySQL80 - W...

File Edit View Query Database Server Tools Scripting Help

Navigator: Project File1 Department file 2 Institute File 3 Course file 4 User file5 practice58 final file 6 Munna1 Munna2 SQL File 11\* x

**MANAGEMENT**

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

**INSTANCE**

- Startup / Shutdown
- Server Logs
- Administration Schemas

No object selected

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

passenger_id	p_name	age	gender	user_id	reservatin_status	ticket_id
501	ramesh	30	M	101	C	1
502	sunita	28	F	103	C	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid Form Editor

```
103
104
105
106 • desc passenger;
107 • insert into passenger values(501,'ramesh',30,'M',101,'C',1),
108          (502,'sunita',28,'F',103,'C',2);
109
110 • select * from passenger;
111
112 -- display user id and name of all those user who booked ticket for duronto express
113 • select user_id,p_name from passenger where ticket_id = 1
```

## station table –

The screenshot shows the MySQL Workbench interface with the SQL editor tab selected. The code pane contains the following SQL script:

```
desc station;
insert into station values(111,'Pune',1,'08:35:00',22204),
                           (222,'Delhi',2,'20:15:00',12267),
                           (333,'Mumbai',0,'06:00:00',12953),
                           (444,'Kolhapur',1,'23:40:00',1951);
select * from station;
```

The Result Grid shows the data inserted into the station table:

No	name	halt	arrival_time	train_no
111	Pune	1	08:35:00	22204
222	Delhi	2	20:15:00	12267
333	Mumbai	0	06:00:00	12953
444	Kolhapur	1	23:40:00	1951

## Ticket table –

The screenshot shows the MySQL Workbench interface with the SQL editor tab selected. The code pane contains the following SQL script:

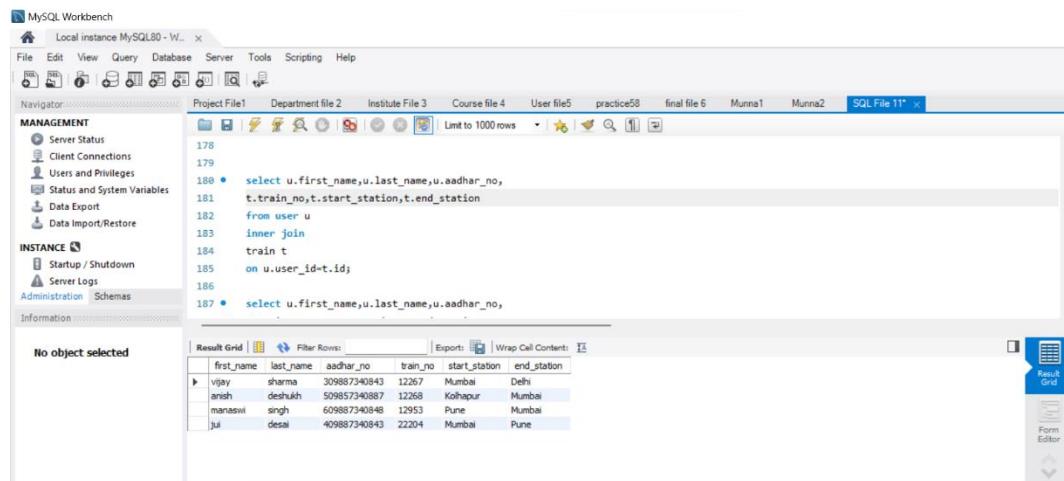
```
desc ticket;
insert into ticket values(1,102,'C',1,12267,15),
                           (2,104,'NC',1,22204,30),
                           (3,101,'C',2,12225,250),
                           (4,102,'NC',1,12951,150);
select * from ticket;
```

The Result Grid shows the data inserted into the ticket table:

id	user_id	status	no_of_passenger	train_no	ticket_price
1	102	C	1	12267	15
2	104	NC	1	22204	30
3	101	C	2	12225	250
4	102	NC	1	12951	150

## Inner Join –

Fetch the detail of railway reservation system (First name, last name , Adhar\_No, train station, start\_station, End\_station) with respect to train to obtain them.



The screenshot shows the MySQL Workbench interface with a SQL editor tab titled "SQL File 11". The query is:

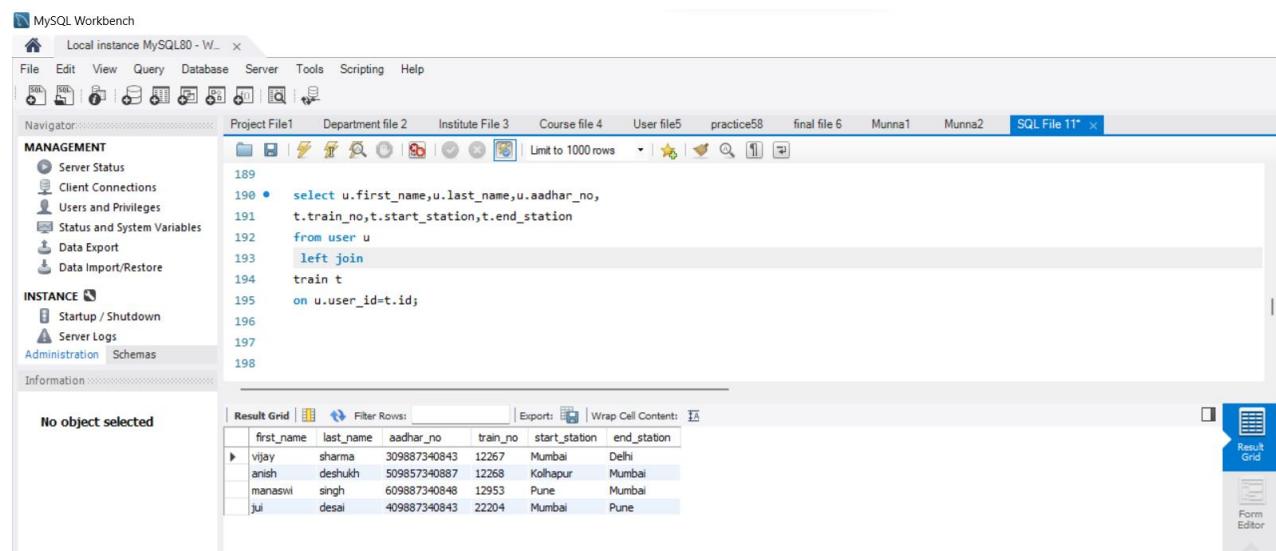
```
188 • select u.first_name,u.last_name,u.aadhar_no,
189     t.train_no,t.start_station,t.end_station
190   from user u
191   inner join
192     train t
193   on u.user_id=t.id;
194
195 • select u.first_name,u.last_name,u.aadhar_no,
```

The results grid displays the following data:

first_name	last_name	aadhar_no	train_no	start_station	end_station
vijay	sharma	309887340843	12267	Mumbai	Delhi
anish	deshukh	509857340887	12268	Kolhapur	Mumbai
manaswi	singh	609887340848	12953	Pune	Mumbai
jui	desai	409887340843	22204	Mumbai	Pune

## Left Join –

Fetch the detail of all user (Name, Place, train\_Station, end\_Station) and First\_name, adhar\_no used to get them.



The screenshot shows the MySQL Workbench interface with a SQL editor tab titled "SQL File 11". The query is:

```
189
190 • select u.first_name,u.last_name,u.aadhar_no,
191     t.train_no,t.start_station,t.end_station
192   from user u
193   left join
194     train t
195   on u.user_id=t.id;
```

The results grid displays the following data:

first_name	last_name	aadhar_no	train_no	start_station	end_station
vijay	sharma	309887340843	12267	Mumbai	Delhi
anish	deshukh	509857340887	12268	Kolhapur	Mumbai
manaswi	singh	609887340848	12953	Pune	Mumbai
jui	desai	409887340843	22204	Mumbai	Pune

## Right Join –

Write a query to display the First\_name, last\_name, adhar\_no, train\_no, start\_station, end\_station table.

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
198
199
200 • select u.first_name,u.last_name,u.aadhar_no,
201     t.train_no,t.start_station,t.end_station
202     from user u
203     right join
204     train t
205     on u.user_id=t.id;
206
207
```

The results grid displays the following data:

first_name	last_name	aadhar_no	train_no	start_station	end_station
vijay	sharma	309887340843	12267	Mumbai	Delhi
anish	deshukh	509857340887	12268	Kolhapur	Mumbai
			12457	mumbai	chennai
			12951	Kolhapur	Mumbai
manaswi	singh	609887340848	12953	Pune	Mumbai
jui	desai	409887340843	22204	Mumbai	Pune

## Join –

write a query to display the common contain of the two table .

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
169 -- join
170 • select u.first_name,u.last_name,u.aadhar_no,
171     t.train_no,t.start_station,t.end_station
172     from user u
173     join
174     train t
175     on u.user_id=t.id;
176
177
178
```

The results grid displays the following data:

first_name	last_name	aadhar_no	train_no	start_station	end_station
vijay	sharma	309887340843	12267	Mumbai	Delhi
anish	deshukh	509857340887	12268	Kolhapur	Mumbai
manaswi	singh	609887340848	12953	Pune	Mumbai
jui	desai	409887340843	22204	Mumbai	Pune

# Full Outer Join

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
-- full outer join
-- (select u.first_name,u.last_name,u.aadhar_no,
-- t.train_no,t.start_station,t.end_station
-- from user u
-- left join
-- train t
-- on u.user_id=t.id)
union
-- (select u.first_name,u.last_name,u.aadhar_no,
-- t.train_no,t.start_station,t.end_station
-- from user u
-- right join
-- train t
-- on u.user_id=t.id);
```

The result grid displays the following data:

first_name	last_name	aadhar_no	train_no	start_station	end_station
vijay	sharma	309887340845	12267	Mumbai	Delhi
prash	deshlak	309887340847	12268	Kolhapur	Mumbai
mukund	singh	609887340848	12953	Pune	Mumbai
JM	desai	409887340843	22204	Mumbai	Pune
1008	1008	1008	12225	mumbai	rathnagiri
1009	1009	1009	12457	mumbai	chennai
1010	1010	1010	12951	Kolhapur	Mumbai

# Order By –

display all the records as per the train\_station whose train\_station is "Mumbai"

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

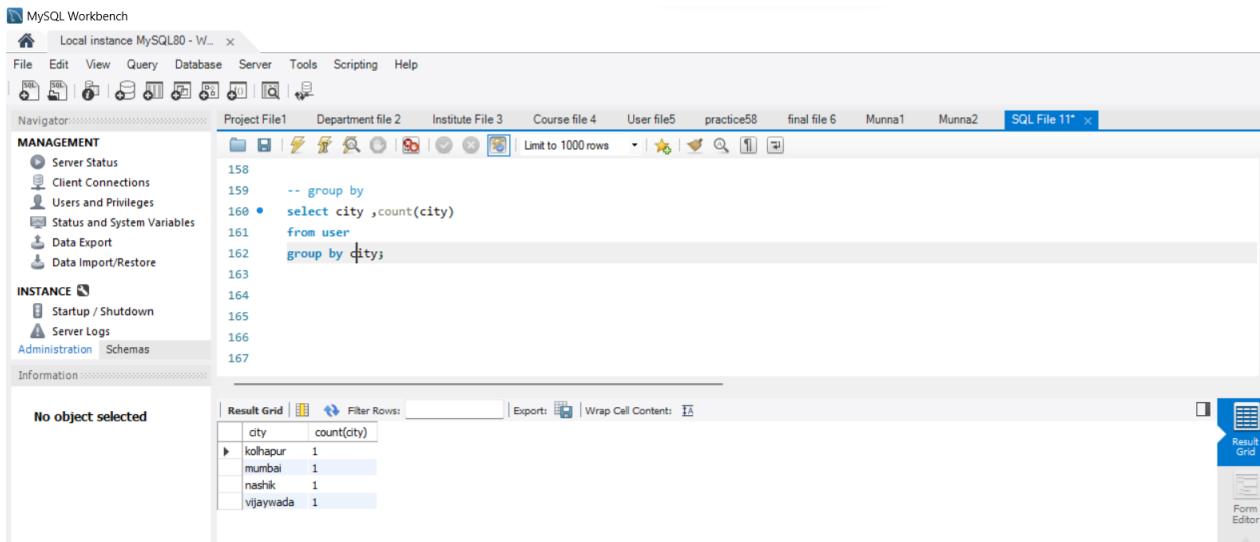
```
-- order by
select * from train
where start_station='mumbai'
order by start_station;
```

The result grid displays the following data:

id	train_no	train_name	arrival_time	departure_time	Start_Station	End_Station
107	12225	MAIL	09:30:00	17:00:00	mumbai	rathnagri
101	12267	duronto express	08:35:00	16:35:00	Mumbai	Delhi
106	12457	MAIL	04:15:00	08:30:00	mumbai	chennai
102	22204	vande bharat express	06:35:00	20:15:00	Mumbai	Pune

## Group By-

display the city and city count according to City



The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query is:

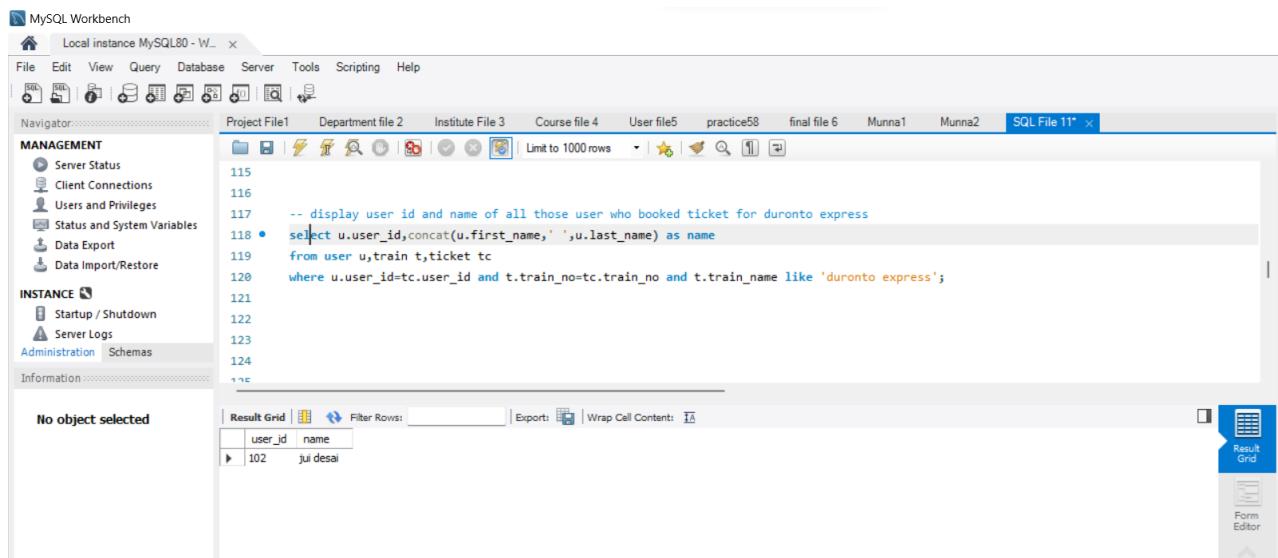
```
158
159 -- group by
160 • select city ,count(city)
161   from user
162   group by city;
163
164
165
166
167
```

The results grid shows the following data:

city	count(city)
kolhapur	1
mumbai	1
nashik	1
vijaywada	1

## DQL - QUERIES

Display user id and name of all those user who booked ticket for Durando express



The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query is:

```
115
116
117 -- display user id and name of all those user who booked ticket for duronto express
118 • select u.user_id,concat(u.first_name,' ',u.last_name) as name
119   from user u,train t,ticket tc
120   where u.user_id=tc.user_id and t.train_no=tc.train_no and t.train_name like 'duronto express';
121
122
123
124
125
```

The results grid shows the following data:

user_id	name
102	jui desai

## Like-

Write a query to display the record of the User whose name start with “R”

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
353
354
355
356
357
358
359
360
361    -- like
362    -- like operator
363 •   select * from User
364     where First_name like 'A%';
365
366
367
368
369
370
371
372
373 •   select * from User
374     where first_name like 'A%';
375
376
377
```

The result grid shows one row of data:

user_id	first_name	last_name	aadhar_no	gender	age	mobile_no	city	state
103	anish	deshukh	509857340887	M	29	9823142314	mumbai	maharashtra

Write a query to display the record of the User whose name end with “R”

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
364     where First_name like 'A%';
365
366
367
368
369
370
371
372
373 •   select * from User
374     where first_name like 'A%';
375
376
377
```

The result grid shows one row of data:

user_id	first_name	last_name	aadhar_no	gender	age	mobile_no	city	state
103	anish	deshukh	509857340887	M	29	9823142314	mumbai	maharashtra

Write a query to display the record of the User whose name contain P letter in the name “%P%”

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
373
374
375
376
377
378
379 •   select * from User
380     where Last_name like "%k%";
```

The result grid shows two rows of data:

user_id	first_name	last_name	aadhar_no	gender	age	mobile_no	city	state
102	jul	desai	409887340843	F	25	950043322	kolhapur	maharashtra
104	manaswi	singh	609887340848	F	27	8678154887	nashik	maharashtra

Write query to display the course whose Course\_Fee between 11500 and 50000

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:`301 -- Range Operator  
302 • select * from Course where Course_fee between 11500 and 50000;  
303  
304  
305  
306`

The results grid displays the following data:

Sl_No	Course_Id	Course_Name	Course_fee	Duration	Department_Name	Class_Name
5	C105	Construction Management	50000	6 Month	Civil	Odin School
6	C106	Structural Design	40000	6 Month	Civil	Odin School
7	C107	Autocad	30000	6 Month	Mechanical	Ethive
9	C109	Crafting	50000	6 Month	Art	Kiran Academy
10	C110	Drawing	20000	6 Month	Art	Kiran Academy
12	C12	MPSG	50000	6 Month	Civil Service	Khan Academy
*	HULL	HULL	HULL	HULL	HULL	HULL

Write a query to display the Course whose Course fees (50000,30000,40000)

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:`303  
304  
305 -- list operator  
306  
307 • select * from Course where Course_fee in (50000,30000,40000);  
308  
309  
310  
311`

The results grid displays the following data:

Sl_No	Course_Id	Course_Name	Course_fee	Duration	Department_Name	Class_Name
5	C105	Construction Management	50000	6 Month	Civil	Odin School
6	C106	Structural Design	40000	6 Month	Civil	Odin School
7	C107	Autocad	30000	6 Month	Mechanical	Ethive
9	C109	Crafting	50000	6 Month	Art	Kiran Academy
12	C12	MPSG	50000	6 Month	Civil Service	Khan Academy
*	HULL	HULL	HULL	HULL	HULL	HULL

The Action Output pane shows the following log entries:

#	Time	Action	Message	Duration / Fetch
118	18:18:51	select * from Course where Course_fee between 10000 and 75000 LIMIT 0, 1000	0 row(s) returned	0.015 sec / 0.000 sec
119	18:19:04	select * from Course where Course_fee between 10000 and 50000 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
120	18:19:13	select * from Course where Course_fee between 11500 and 50000 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
121	18:19:19	select * from Course where Course_fee between 11500 and 50000 LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
122	18:24:17	select * from Course where Course_fee in (50000,30000,40000) LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Write a query to display the all unique start\_station From ticket

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons for database management. The left sidebar has sections for MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore) and INSTANCE (Administration, Schemas, Information). The main area shows a query editor with the following SQL code:

```
419  
420  
421  
422  
423    -- Distinct  
424 ● select * from Institute;  
425 ● select Distinct start_station from train;  
426  
427  
428  
429  
430  
431  
432
```

Below the code is a results grid titled "Result Grid". It has a header row with "start\_station" and "mumbai". The data row shows "Kolhapur" and "Pune". There are buttons for "Filter Rows:", "Export:", and "Wrap Cell Content:".

Write a query to display the start\_station which is not assign to any User

Write a query to display the train name who have enter the start\_station

The screenshot shows the MySQL Workbench interface. The left sidebar has sections for MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore) and INSTANCE (Administration, Schemas). The central area contains a SQL editor with the following code:

```
433
434
435
436
437 -- Is Null and Not Null
438 • select * from User;
439 • select * from train where start_station is Null;
440 • select * from train where start_station is not Null;
441
442
443
444
445
446
```

Below the code is a Result Grid showing the results of the query:

ID	train_no	train_name	arrival_time	departure_time	start_station	end_station
107	12225	kalpvriksh	09:30:00	17:00:00	mumbai	rathnagiri
101	12267	duronto express	08:35:00	16:35:00	Mumbai	Delhi
103	12286	mahalaxmi express	09:00:00	23:40:00	Kolhapur	Kolhapur
108	12291	kalpvriksh	08:30:00	16:30:00	mumbai	rathnagiri
105	12951	sahyadri express	08:35:00	16:35:00	Kolhapur	Mumbai
104	12953	deccan express	10:55:00	17:40:00	Pune	Mumbai
102	22204	vande bharat express	06:35:00	20:15:00	Mumbai	Pune

## Built in SQL – String Functions

The screenshot shows the MySQL Workbench interface. The left sidebar has sections for MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore) and INSTANCE (Startup / Shutdown, Server Logs). The central area contains a SQL editor with the following code:

```
139 -- display project name in capitalized manner
140 • select lower('RAILWAY MANAGEMENT SYSTEM');
141 • select upper('railway management system');
142 • select substr('railway managemet',1,7);
143 • select replace('railway management','management','system');
144
145
146
```

Write a query to Display the “railway management system” in Lower case

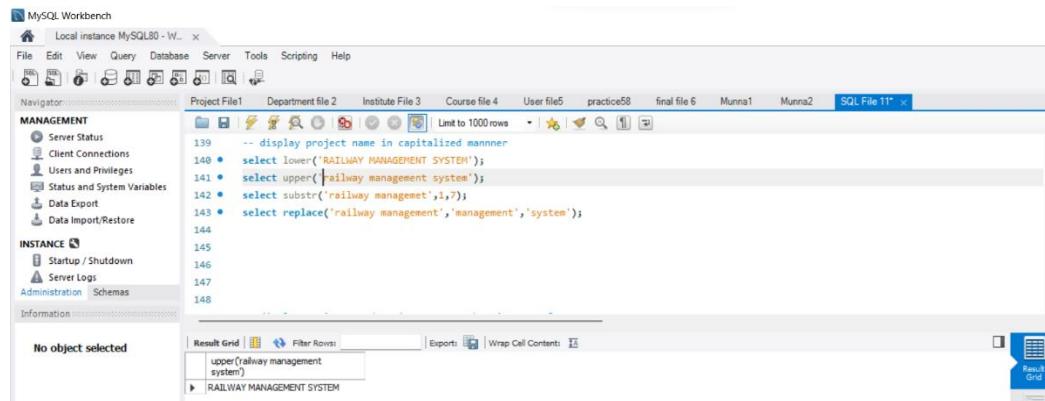
The screenshot shows the MySQL Workbench interface. The left sidebar has sections for MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore) and INSTANCE (Startup / Shutdown, Server Logs). The central area contains a SQL editor with the following code:

```
139 -- display project name in capitalized manner
140 • select lower('RAILWAY MANAGEMENT SYSTEM');
141 • select upper('railway management system');
142 • select substr('railway managemet',1,7);
143 • select replace('railway management','management','system');
144
145
146
147
148
```

Below the code is a Result Grid showing the results of the query:

lower(RAILWAY MANAGEMENT SYSTEM)
railway management system

Write a query to Display the “railway management system” in capitalize manner

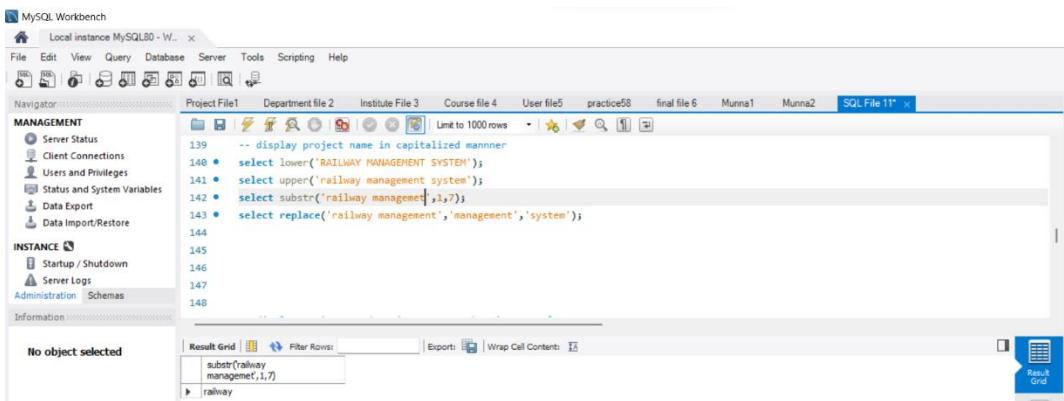


The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```
-- display project name in capitalized manner
select lower('RAILWAY MANAGEMENT SYSTEM');
select upper('railway management system');
select substr('railway managemet',1,7);
select replace('railway management','management','system');
```

The result grid shows the output of the first query: RAILWAY MANAGEMENT SYSTEM.

Write a query to Display the “railway management system” only with “railway”

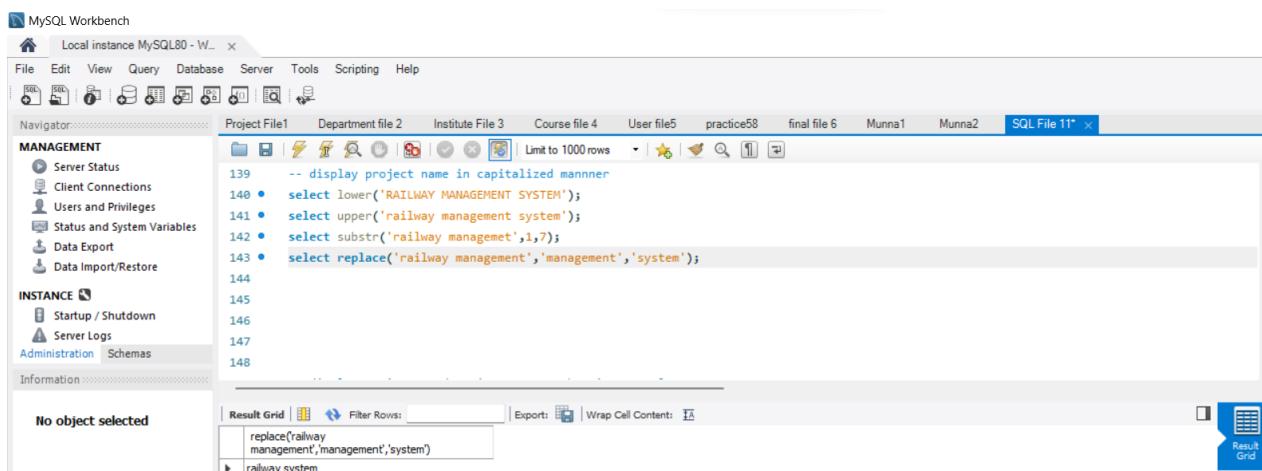


The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```
-- display project name in capitalized manner
select lower('RAILWAY MANAGEMENT SYSTEM');
select upper('railway management system');
select substr('railway managemet',1,7);
select replace('railway management','management','system');
```

The result grid shows the output of the third query: substr(railway managemet,1,7) and railway.

Write a query to replace the “management” word to “system”.

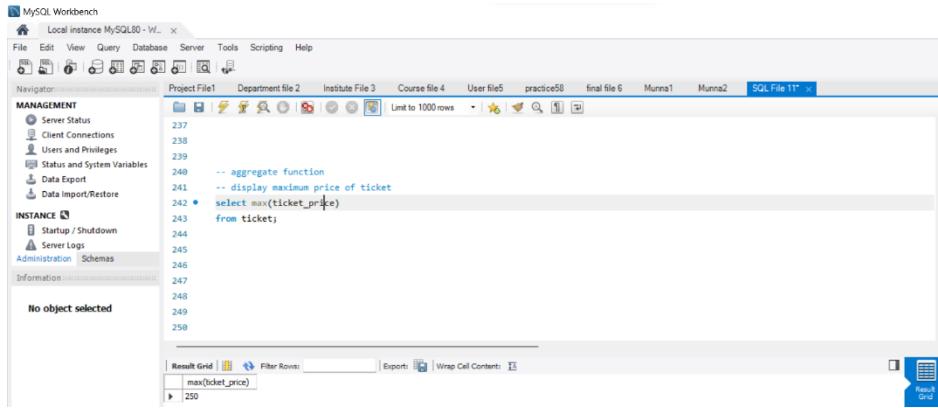


The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```
-- display project name in capitalized manner
select lower('RAILWAY MANAGEMENT SYSTEM');
select upper('railway management system');
select substr('railway managemet',1,7);
select replace('railway management','management','system');
```

The result grid shows the output of the fourth query: replace(railway management,'management','system') and railway system.

## Aggregate Function -

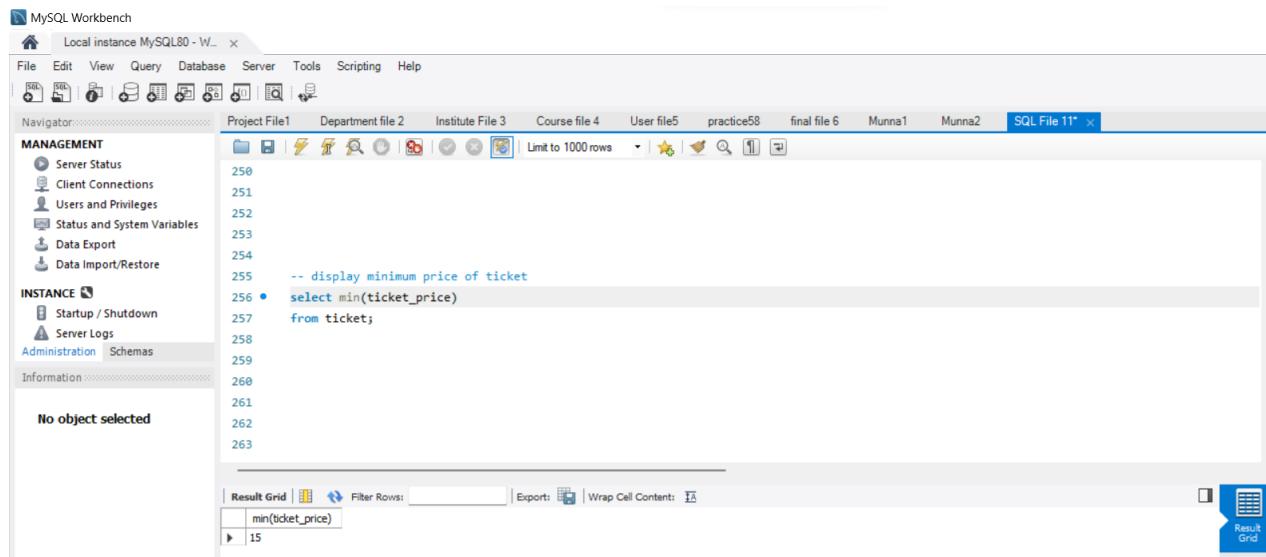


The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```
-- aggregate function
-- display maximum price of ticket
select max(ticket_price)
from ticket;
```

The result grid shows a single row with the value 250.

Write a query to display the minimum train\_price



The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```
-- display minimum price of ticket
select min(ticket_price)
from ticket;
```

The result grid shows a single row with the value 15.

Write a query to display the Avg train\_price

The screenshot shows the MySQL Workbench interface. The SQL editor tab is active, containing the following SQL code:

```
260
261
262
263
264
265    -- display average of ticket
266 •   select avg(ticket_price)
267     from ticket;
268
269
270
271
272
273
```

The result grid below shows the output of the query:

avg(ticket_price)
111.2500

Write a query to display the sum of the train\_price

The screenshot shows the MySQL Workbench interface. The SQL editor tab is active, containing the following SQL code:

```
269
270
271
272
273
274
275    -- display sum of ticket
276 •   select sum(ticket_price)
277     from ticket;
278
279
280
281
282
```

The result grid below shows the output of the query:

sum(ticket_price)
445

## Subqueries –

Write a query to display train no and train name whose arrival time is less than Sahyadri express

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The code is as follows:

```
209  
210  
211  
212  
213  
214  
215  
216 -- subqueries  
217 -- write a query to display train_no & train_name whose arrival time is less than sahyadri express  
218 • select train_no,train_name  
from train  
where arrival_time < (select arrival_time from train  
where train_name = "sahyadri express");
```

The result grid shows one row with both columns set to NULL.

Write a query to show second lowest price of train ticket

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The code is as follows:

```
224  
225  
226  
227  
228  
229 -- write a query to show second lowest price of train ticket  
230 • select min(ticket_price) from ticket  
where ticket_price > (select min(ticket_price) from ticket);  
231  
232  
233  
234  
235  
236
```

The result grid shows one row with the value 30.

Write a query to display ticket price whose price is less than average price

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The code is as follows:

```
232  
233  
234  
235  
236 -- write a query to display ticket price whose price is less than average price  
237 • select ticket_price from ticket  
where ticket_price < (select avg(ticket_price) from ticket);  
238  
239  
240  
241  
242  
243  
244  
245
```

The result grid shows two rows with values 15 and 30.