

**VISVESVARAYA TECHNOLOGICAL
UNIVERSITY**

“JnanaSangama”, Belgaum -590014, Karnataka.



**LAB REPORT
on**

Database Management Systems (23CS3PCDBM)

Submitted by

PRAJWAL S

(1BM24CS209)

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

Sep-2025 to Jan-2026

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 work entitled “Database Management Systems (23CS3PCDBM)” carried out by **PRAJWAL S(1BM24CS209)**, who is bonafide student 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 Visvesvaraya Technological University, Belgaum during the year 2025-2026. The Lab report has been approved as it satisfies the academic requirements in respect of a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

Ms. Rashmi H Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE
---	--

Index

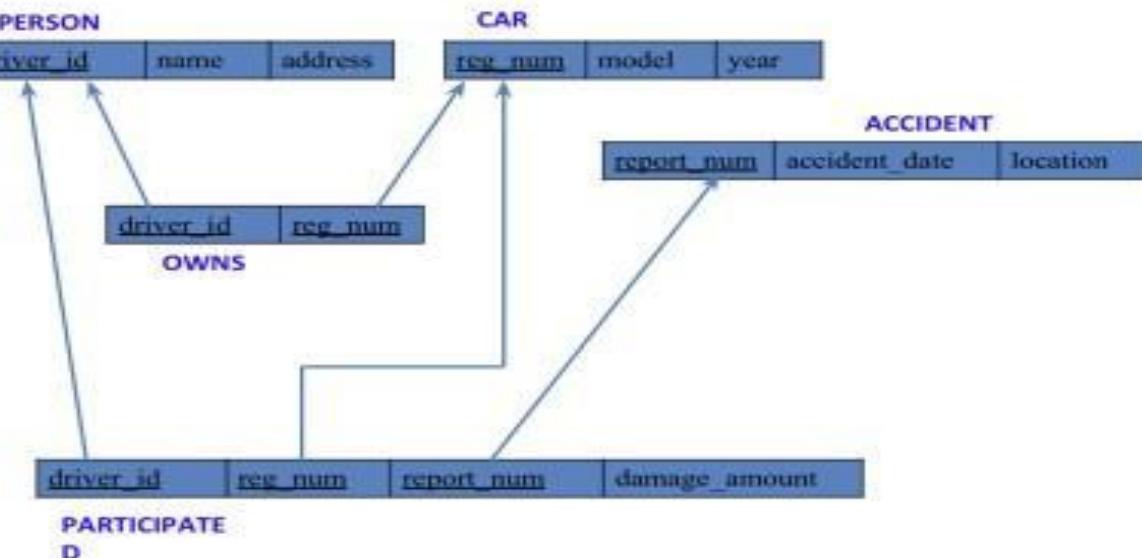
Sl. No.	Experiment Title	Page No.
1	Insurance Database	4-12
2	More Queries on Insurance Database	13-15
3	Bank Database	16-25
4	More Queries on Bank Database	26-27
5	Employee Database	28-37
6	More Queries on Employee Database	38-39
7	Supplier Database	40-46
8	More Queries on Supplier Database	47-49
9	NOSQL Installation in Cloud	50
10	NO SQL - Student Database	51-52
11	NO SQL - Customer Database	53-55
12	NO SQL – Restaurant Database	56-57

Experiment 1: Insurance Database

Specification of Insurance Database Application

The insurance database must maintain information about drivers, the cars they own, the accidents reported, and the participation of each driver and car in those accidents. Each driver in the system is uniquely identified by a driver ID, along with their name and address, and each car is uniquely identified by its registration number together with details such as model and manufacturing year. The system must allow storing ownership information that links a driver to one or more cars, while also allowing a car to be linked to one or more drivers if shared ownership occurs; duplicate ownership records for the same driver and car must not exist. Accident information must be stored using a unique report number assigned to each accident, along with the date on which the accident occurred and the location where it happened. Every accident reported in the system must have at least one participating driver and car, and this participation is recorded by linking the driver, the involved car, and the accident report together with the corresponding damage amount for that particular involvement. A participation record must reference an existing driver, an existing car, and an existing accident, and no two participation entries may repeat the same combination of driver, car, and accident report. The database must ensure that damage amounts are non-negative, accident dates are valid calendar dates, and car manufacturing years fall within reasonable limits. It must also preserve referential integrity so that ownership or participation entries cannot exist without valid driver, car, and accident information already present in the system. Deletion policies must prevent removal of drivers or cars that appear in past accident participation records unless historical consistency is preserved through controlled deletion rules or archival mechanisms. The system should maintain accurate links between drivers, cars, and accidents at all times, ensuring reliable retrieval of ownership histories, accident histories, and damage information for administrative, legal, and insurance-related purposes.

Schema Diagram



Create database

```
CREATE DATABASE IF NOT EXISTS INSURANCE ;
SHOW DATABASES ;
USE INSURANCE ;
```

Create table

```
CREATE TABLE IF NOT EXISTS PERSON
(   driver_id VARCHAR(10) PRIMARY
KEY,   name VARCHAR(50) NOT NULL,
address VARCHAR(100)
);
CREATE TABLE IF NOT EXISTS CAR (
reg_num VARCHAR(10) PRIMARY KEY,
model VARCHAR(20),
year INT
);
CREATE TABLE IF NOT EXISTS ACCIDENT (
report_num INT PRIMARY KEY,
accident_date DATE,
location VARCHAR(50)
);
CREATE TABLE IF NOT EXISTS OWNS
(   driver_id VARCHAR(10),   reg_num
VARCHAR(10),
```

```

PRIMARY KEY (driver_id, reg_num),
FOREIGN KEY (driver_id) REFERENCES PERSON(driver_id),
FOREIGN KEY (reg_num) REFERENCES CAR(reg_num)
);
CREATE TABLE IF NOT EXISTS PARTICIPATED
( driver_id VARCHAR(10), reg_num
VARCHAR(10), report_num INT,
damage_amount INT,
PRIMARY KEY (driver_id, reg_num, report_num),
FOREIGN KEY (driver_id) REFERENCES PERSON(driver_id),
FOREIGN KEY (reg_num) REFERENCES CAR(reg_num),
FOREIGN KEY (report_num) REFERENCES ACCIDENT(report_num)
);

```

Structure of the table

desc person;

	Field	Type	Null	Key	Default	Extra
▶	driver_id	varchar(20)	NO	PRI	NULL	
	reg_num	varchar(10)	NO	PRI	NULL	
	report_num	int	NO	PRI	NULL	
	damage_amount	int	YES		NULL	

desc accident;

	Field	Type	Null	Key	Default	Extra
▶	report_num	int	NO	PRI	NULL	
	accident_date	date	YES		NULL	
	location	varchar(50)	YES		NULL	

```
desc participated;
```

Field		Type	Null	Key	Default	Extra
▶	driver_id	varchar(20)	NO	PRI	NULL	
	reg_num	varchar(10)	NO	PRI	NULL	
	report_num	int	NO	PRI	NULL	
	damage_amount	int	YES		NULL	

```
desc car;
```

Field		Type	Null	Key	Default	Extra
▶	reg_num	varchar(15)	NO	PRI	NULL	
	model	varchar(10)	YES		NULL	
	year	int	YES		NULL	

```
desc owns;
```

Field		Type	Null	Key	Default	Extra
▶	driver_id	varchar(20)	NO	PRI	NULL	
	reg_num	varchar(10)	NO	PRI	NULL	

Inserting Values to the table

```
insert into person values("A01","Richard", "Srinivas nagar");
insert into person values("A02","Pradeep", "Rajaji nagar");
insert into person values("A03","Smith", "Ashok nagar");
insert into person values("A04","Venu", "N R Colony"); insert
into person values("A05","John", "Hanumanth nagar"); select
* from person;
```

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

	driver_id	name	address
▶	A01	Richard	Srinivas nagar
	A02	Pradeep	Rajaji nagar
	A03	Smith	Ashok nagar
	A04	Venu	N R. Colony
	A05	John	Manumanth nagar

person 19 ×

```
insert into car values("KA052250","Indica", "1990");
insert into car values("KA031181","Lancer", "1957");
insert into car values("KA095477","Toyota", "1998");
insert into car values("KA053408","Honda", "2008");
insert into car values("KA041702","Audi", "2005"); select
* from car;
```

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

	reg_num	model	year
▶	KA031181	Lancer	1957
	KA041702	Audi	2005
	KA052250	Indica	1990
	KA053408	Honda	2008
	KA095477	Toyota	1998

car 20 ×

```
insert into owns values("A01","KA052250"); insert into owns values("A02","KA031181");
insert into owns values("A03","KA095477"); insert into owns values("A04","KA053408");
insert into owns values("A05","KA041702"); select * from owns;
```

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

	driver_id	reg_num
▶	A02	KA031181
	A05	KA041702
	A01	KA052250
	A04	KA053408
	A03	KA095477

owns 22 ×

```
insert into accident values(11,'2003-01-01',"Mysore Road"); insert
into accident values(12,'2004-02-02',"South end Circle"); insert
into accident values(13,'2003-01-21',"Bull temple Road"); insert
into accident values(14,'2008-02-17',"Mysore Road"); insert into
accident values(15,'2004-03-05',"Kanakpura Road");
```

```
select * from accident;
```

Result Grid				Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	report_num	accident_date	location				
▶	11	2003-01-01	Mysore Road				
	12	2004-02-02	South end Circle				
	13	2003-01-21	Bull temple Road				
	14	2008-02-17	Mysore Road				
	15	2004-03-05	Kanakpura Road				

accident 23 x

```
insert into participated values("A01","KA052250",11,10000);
insert into participated values("A02","KA053408",12,50000);
insert into participated values("A03","KA095477",13,25000);
insert into participated values("A04","KA031181",14,3000); insert
into participated values("A05","KA041702",15,5000); select *
from participated;
```

Result Grid				Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:	
	driver_id	reg_num	report_num	damage_amount				
▶	A01	KA052250	11	10000				
	A02	KA053408	12	25000				
	A03	KA095477	13	25000				
	A04	KA031181	14	3000				
	A05	KA041702	15	5000				

participated 24 x

Queries :-

Display Accident date and location

```
66
67
68 •   select accident_date,location from accident;
69
70
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	accident_date	location
▶	2003-01-01	Mysore road
	2004-02-02	South end Circle
	2003-01-21	Bull temple Road
	2008-02-17	Mysore road
	2005-03-04	Kanakpura Road

Update the damage amount to 25000 for the car with a specific reg_num (example 'KA053408') for which the accident report number was 12.

```
59
60 •   update participated set damage_amount=25000 where reg_num="KA053408" and report_num=12;
61
62 •   select * from participated;
63
```

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

	driver_id	reg_num	report_num	damage_amount
▶	A01	KA052250	11	10000
	A02	KA053408	12	25000
	A03	KA095477	13	25000
	A04	KA031181	14	3000
	A05	KA041702	15	5000
*	HULL	HULL	HULL	HULL

Display Accident date and location

```

45 •    select * from accident;
46
47 •    create table participated(driver_id varchar(10), reg_num varchar(10),
48      report_num int, damage_amount int, primary key(driver_id, reg_num, report_num),
49      foreign key(driver_id) references person(driver_id),
50      foreign key(reg_num) references car(reg_num), foreign key(report_num) references accident(report_num));
51
52 •    insert into participated values("A01","KA052250",11,10000);
53 •    insert into participated values("A02","KA053408",12,50000);
54 •    insert into participated values("A03","KA095477",13,25000);
55 •    insert into participated values("A04","KA031181",14,3000);

```

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

	report_num	accident_date	location
▶	11	2003-01-01	Mysore road
	12	2004-02-02	South end Circle
	13	2003-01-21	Bull temple Road
	14	2008-02-17	Mysore road
●	15	2005-03-04	Kanakpura Road
●	HULL	HULL	HULL

Display driver id who did accident with damage amount greater than or equal to Rs.25000

```

69
70
71 •    select driver_id from participated p, accident a where p.report_num=a.report_num and damage_amount>=25000;
72
73

```

Result Grid | Filter Rows: _____ | Export: _____ | Wrap Cell Content: _____

	driver_id
▶	A02
	A03

Experiment 2: More Queries on Insurance Database

Queries :-

Display the entire CAR relation in the ascending order of manufacturing year.

```
78  
79  
80 •  select * from car order by year asc;  
81  
82
```

	reg_num	model	year
▶	KA031181	Lancer	1957
	KA052250	Indica	1990
	KA095477	Toyota	1998
	KA041702	Audi	2005
*	KA053408	Honda	2008
	NULL	NULL	NULL

Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved

```
82  
83 •  select count(model) CNT from car c, participated p, accident a where c.model="Lancer" and c.reg_num=p.reg_num  
84      and p.report_num=a.report_num;  
85  
86
```

	CNT
▶	1

Find the Average Damage Amount

```
75  
76  
77 •   select avg(damage_amount) from participated;  
78  
79
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	
avg(damage_amount)					
▶	13600.0000				

Delete the tuple whose Damage Amount is below the Average Damage Amount

```
86  
87 • ⚡ delete from participated where damage_amount < (select avg_damage from (select avg(damage_amount) as  
88     avg_damage from participated)as t);  
89  
90 •   select * from participated;
```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
driver_id	reg_num	report_num	damage_amount		
▶ A02	KA053408	12	25000		
A03	KA095477	13	25000		
*	NULL	NULL	NULL		

List the name of drivers whose Damage is Greater than the Average Damage Amount

```
91  
92  
93 •   select name from person a, participated b where a.driver_id = b.driver_id and damage_amount >  
94     (select avg(damage_amount) from participated);  
95  
--
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	name			
▶	Pradeep Smith			

Find Maximum Damage Amount.

```
92  
93 •   select name from person a, participated b where a.driver_id = b.driver_id and damage_amount >  
94     (select avg(damage_amount) from participated);  
95  
96  
97 •   select max(damage_amount) from participated;
```

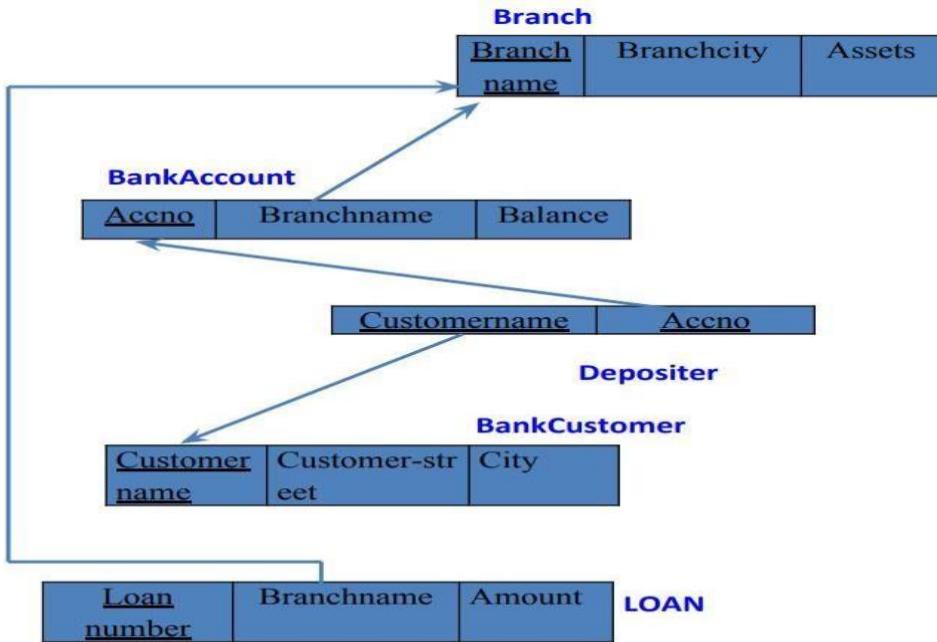
Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	max(damage_amount)			
▶	25000			

Experiment 3: BANKING DATABASE

The banking system must store information about branches, bank accounts, customers, deposit ,relationships, and loans so that branch details (identified by branch name together with city and total assets) are linked to accounts and loans, each account (identified by an account number) records the branch it belongs to and the current balance, customers are recorded with their name, street , city and a depositor relationship associates a customer with an account; loans are recorded by a unique loan number together with the branch name that issued the loan and the loan amount. Account numbers and loan numbers must be unique identifiers, branch names are used to associate accounts and loans to a branch, and customer names (as model) are used to identify customers referenced by depositor entries; every depositor entry must reference an existing customer and an existing account so that ownership and access relationships are always valid, and duplicate depositor records linking the same customer and account are disallowed. The system must maintain referential integrity so accounts cannot reference a non-existent branch, depositor rows cannot reference missing customers or accounts, and loans must reference an existing branch; deletion of a branch, account, or customer that is referenced by dependent records should be controlled (either disallowed or handled by archival/controlled reassignment) to preserve historical transaction and loan consistency. Numeric and temporal constraints must be enforced: account balances should be constrained to valid values (for example non-negative where overdraft is not allowed), branch assets and loan amounts must be non- negative and within specified business limits, and updates to balance or loan amounts should be auditable. Cardinality rules implied by the schema are enforced: a branch may host many accounts and issue many loans, an account belongs to exactly one branch, a customer may be linked to many accounts through depositor relationships, and an account may have many depositors if joint accounts are permitted by policy. Implementation must prevent orphaned records, ensure uniqueness where required, and rely on application logic or database-level triggers to enforce complex rules such as cascading effects on deletion, business rules about allowed balance operations or overdrafts, and any required validation when transferring accounts between branches or when converting a customer's identifying details; the database should thus reliably support queries for branch-wise account lists, customer account

ownership, account balances, and loan portfolios while preserving historical and referential integrity for auditing and regulatory reporting.

Schema Diagram



Create Database

```
create database bank_database;
use bank_database;
```

Create Table

```
create table branch(
branch_name varchar(50),
branch_city varchar(50),
assests real, primary
key(branch_name)
);
```

```
create table bankAccount( accno int, branch_name
varchar(50), balance real, primary key (accno), foreign
key(branch_name) references branch(branch_name)
);
```

```
create table bankCustomer(
customer_name varchar(50),
customer_street varchar(50),
customer_city varchar(50) ,
primary key (customer_name)
);
```

```
create table depositor(
customer_name varchar(50),
accno int,
primary key (customer_name,accno), foreign key(customer_name)
references bankCustomer(customer_name), foreign key(accno)
references bankAccount(accno)
);
desc depositor;
```

```
create table loan( loan_number int, branch_name
varchar(50), amount real, primary key(loan_number),
foreign key(branch_name) references
branch(branch_name)
);
```

Structure of the table

desc branch;

	Field	Type	Null	Key	Default	Extra
▶	branch_name	varchar(30)	NO	PRI	NULL	
	branch_city	varchar(30)	YES		NULL	
	assets	double	YES		NULL	

desc bankaccount;

	Field	Type	Null	Key	Default	Extra
▶	accno	int	NO	PRI	NULL	
	branch_name	varchar(30)	YES	MUL	NULL	
	balance	double	YES		NULL	

desc loan;

	Field	Type	Null	Key	Default	Extra
▶	loan_number	int	NO	PRI	NULL	
	branch_name	varchar(30)	YES	MUL	NULL	
	amount	double	YES		NULL	

desc bankcustomer;

	Field	Type	Null	Key	Default	Extra
▶	customer_name	varchar(30)	NO	PRI	NULL	
	customer_street	varchar(30)	YES		NULL	
	customer_city	varchar(30)	YES		NULL	

desc depositer;

Result Grid		Filter Rows:	Export:		Wrap Cell Content	
	Field	Type	Null	Key	Default	Extra
▶	customer_name	varchar(30)	NO	PRI	NULL	
	accno	int	NO	PRI	NULL	

Insertion of values into tables

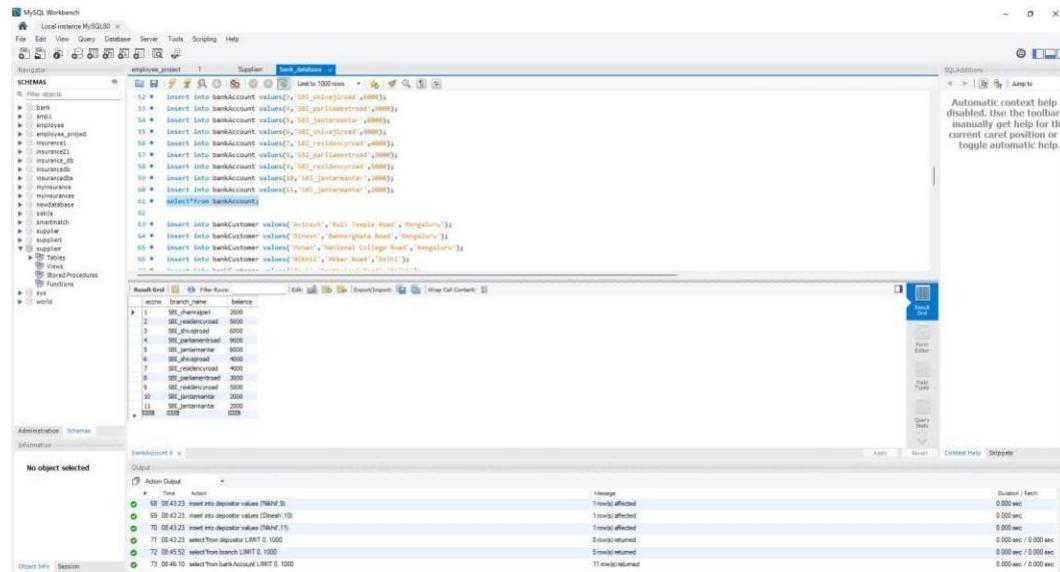
```
insert into branch values('SBI_chamrajpet','bangalore',50000);
insert into branch values('SBI_residencyroad','bangalore',10000);
insert into branch values('SBI_shivajiroad','bombay',20000); insert
into branch values('SBI_parliamentroad','delhi',10000); insert into
branch values('SBI_jantarmantar','delhi',20000); select*from
branch;
```

The screenshot shows the MySQL Workbench interface. On the left, the schema browser displays the 'branch' table under the 'employee_project' schema. On the right, the results grid shows the following data:

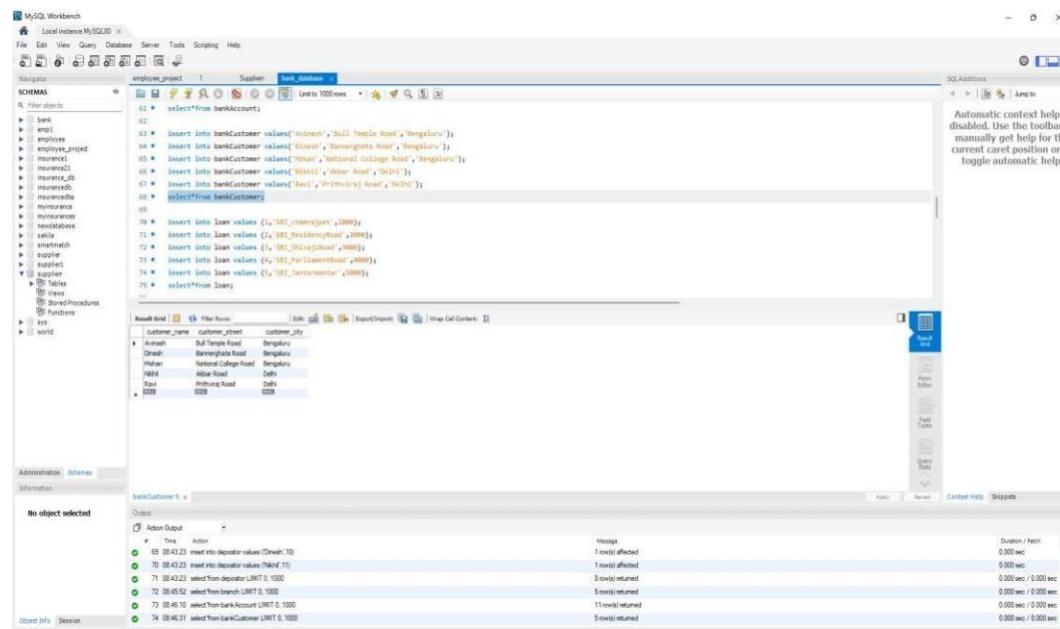
branch_id	branch_name	branch_city	assets
1	SBI_chamrajpet	bangalore	50000
2	SBI_residencyroad	bangalore	10000
3	SBI_shivajiroad	bombay	20000
4	SBI_parliamentroad	delhi	10000
5	SBI_jantarmantar	delhi	20000

```
insert into bankAccount values(1,'SBI_chamrajpet',2000); insert
into bankAccount values(2,'SBI_residencyroad',5000); insert
into bankAccount values(3,'SBI_shivajiroad',6000); insert into
bankAccount values(4,'SBI_parliamentroad',9000); insert into
bankAccount values(5,'SBI_jantarmantar',8000); insert into
bankAccount values(6,'SBI_shivajiroad',4000); insert into
bankAccount values(7,'SBI_residencyroad',4000); insert into
bankAccount values(8,'SBI_parliamentroad',3000); insert into
bankAccount values(9,'SBI_residencyroad',5000); insert into
bankAccount values(10,'SBI_jantarmantar',2000); insert into
```

```
bankAccount values(11,'SBI_jantarmantar',2000); select*from bankAccount;
```



```
insert into bankCustomer values('Avinash','Bull Temple Road','Bengaluru');  
insert into bankCustomer values('Dinesh','Bannerghata Road','Bengaluru');  
insert into bankCustomer values('Mohan','National College  
Road','Bengaluru'); insert into bankCustomer values('Nikhil','Akbar  
Road','Delhi'); insert into bankCustomer values('Ravi','Prithviraj  
Road','Delhi'); select*from bankCustomer;
```



```
insert into loan values (1,'SBI_chamrajpet',1000); insert  
into loan values (2,'SBI_ResidencyRoad',2000); insert  
into loan values (3,'SBI_ShivajiRoad',3000); insert into  
loan values (4,'SBI_ParliamentRoad',4000); insert into  
loan values (5,'SBI_Jantarmantar',5000); select*from  
loan;
```

The screenshot shows the MySQL Workbench interface with several windows open. At the top, there's a menu bar with File, Edit, View, Query, Database, Server, Tools, Scripting, Help. Below the menu is a toolbar with various icons for database management. The main area has a 'Schemas' tree on the left containing 'employees', 'suppliers', 'customers', and 'routines'. A central pane displays a 'Task List' with numerous tasks numbered 67 through 75, each with a brief description like 'Insert into bankCustomer values (...)' or 'select * from bankCustomer'. Below this is a 'Result Grid' showing data for 'customer' and 'bankAccount' tables. On the right, there's a 'Results' pane with tabs for 'Text', 'Binary', 'JSON', and 'Table', and a 'Details' pane below it. The bottom navigation bar includes tabs for 'Object Info', 'Session', 'Context Help', and 'Snippets'.

```
insert into depositor values ('Avinash',1);
insert into depositor values ('Dinesh',2);
insert into depositor values ('Nikhil',4);
insert into depositor values ('Ravi',5);
insert into depositor values ('Avinash',8);
insert into depositor values ('Nikhil',9);
insert into depositor values ('Dinesh',10);
insert into depositor values ('Nikhil',11);

select*from depositor;
```

```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Schemas
employee_project T Supplier bank_database
76 • insert into depositor values ('HDFC','11')
77 • insert into depositor values ('SBI','12')
78 • insert into depositor values ('HBL','13')
79 • insert into depositor values ('Kotak','14')
80 • insert into depositor values ('Axis','15')
81 • insert into depositor values ('ICICI','16')
82 • insert into depositor values ('NBFC1','17')
83 • insert into depositor values ('NBFC2','18')
84 • insert into depositor values ('NBFC3','19')
85 • select branch_name , round(assets/100000,2) as
86 "assets_in_lakhs" from branch
87
88 • select customer_name,branch_name from depositor,
89 bankaccount where depositor.acno = bankAccount.acno
90 group by customer_name,branch_name
91 having count(depositor.acno)>2
92
93 • create view BranchLoanSummary as
94 select branch_name,sum(amount) as total_loan_amount
95
96

```

Result Grid:

branch_name	assets_in_lakhs
SBI_chennai	0.5
SBI_punjab	0.2
SBI_parlamentary	0.1
SBI_residential	0.1
SBI_banglore	0.2

Action Output:

Time	Action	Message	Duration / Fetch
71 08:45:23	select from depositor LIMIT 0, 1000	5rows returned	0.000 sec / 0.000 sec
72 08:45:32	select from branch LIMIT 0, 1000	11rows returned	0.000 sec / 0.000 sec
73 08:46:10	select from bankAccount LIMIT 0, 1000	5rows returned	0.000 sec / 0.000 sec
74 08:46:31	select from bankCustomer LIMIT 0, 1000	5rows returned	0.000 sec / 0.000 sec
75 08:47:04	select from loan LIMIT 0, 1000	8rows returned	0.000 sec / 0.000 sec
76 08:47:16	select from depositor LIMIT 0, 1000	8rows returned	0.000 sec / 0.000 sec
77 08:47:28	select branch_name , round(assets/100000,2) as "assets_in_lakhs" from branch LIMIT 0, 1000	5rows returned	0.000 sec / 0.000 sec

Queries :-

Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.

```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Schemas
employee_project T Supplier bank_database
82 • insert into depositor values ('HDFC','11')
83 • insert into depositor values ('SBI','12')
84 • insert into depositor values ('HBL','13')
85 • select branch_name , round(assets/100000,2) as
86 "assets_in_lakhs" from branch
87
88 • select customer_name,branch_name from depositor,
89 bankaccount where depositor.acno = bankAccount.acno
90 group by customer_name,branch_name
91 having count(depositor.acno)>2
92
93 • create view BranchLoanSummary as
94 select branch_name,sum(amount) as total_loan_amount
95
96

```

Result Grid:

branch_name	assets_in_lakhs
SBI_chennai	0.5
SBI_punjab	0.2
SBI_parlamentary	0.1
SBI_residential	0.1
SBI_banglore	0.2

Action Output:

Time	Action	Message	Duration / Fetch
72 08:45:23	select from depositor LIMIT 0, 1000	5rows returned	0.000 sec / 0.000 sec
73 08:45:32	select from branch LIMIT 0, 1000	11rows returned	0.000 sec / 0.000 sec
74 08:46:10	select from bankAccount LIMIT 0, 1000	5rows returned	0.000 sec / 0.000 sec
75 08:46:31	select from bankCustomer LIMIT 0, 1000	5rows returned	0.000 sec / 0.000 sec
76 08:47:04	select from loan LIMIT 0, 1000	8rows returned	0.000 sec / 0.000 sec
77 08:47:28	select branch_name , round(assets/100000,2) as "assets_in_lakhs" from branch LIMIT 0, 1000	5rows returned	0.000 sec / 0.000 sec

Find all the customers who have at least two accounts at the same branch (ex. 22 | Page

SBI_ResidencyRoad).

The screenshot shows the MySQL Workbench interface with the 'bank_database' selected. In the SQL Editor tab, the following SQL code is written:

```

85 * select*from depositor;
86 *
87 * select branch_name , round(assets/100000,2) as
88 * "assets in lakhs" from branch;
89 *
90 * select customer_name,branch_name from depositor,
91 * bankAccount where depositor.acco = bankAccount.acco
92 * group by customer_name,branch_name
93 * having count(depositor.acco)>=2
94 *
95 * create view BranchLoanSummary as
96 * select branch_name,sum(amount) as total_loan_amount
97 * from loan group by branch_name;
98 *
99 * select * from BranchLoanSummary;

```

The Result Grid shows the output of the last query:

customer_name	branch_name

The Activity Log at the bottom shows the execution history:

Action	Time	Message	Duration / Fetch
73 08:46:10	selected from bankAccount LIMIT 0, 1000	11 rows returned	0.000 sec / 0.000 sec
74 08:46:31	selected from bankCustomer LIMIT 0, 1000	5 rows returned	0.000 sec / 0.000 sec
75 08:47:04	selected from loan LIMIT 0, 1000	5 rows returned	0.000 sec / 0.000 sec
76 08:47:16	selected from depositor LIMIT 0, 1000	3 rows returned	0.000 sec / 0.000 sec
77 08:47:28	select branch_name , round(assets/100000,2) as "assets in lakhs" from branch LIMIT 0, 1000	5 rows returned	0.000 sec / 0.000 sec
78 08:47:37	select customer_name,branch_name from depositor, bankAccount where depositor.acco = bankAccount.acco group by customer_name,branch_name;	0 rows returned	0.000 sec / 0.000 sec

Create a view which gives each branch the sum of all the loans at the branch.

The screenshot shows the MySQL Workbench interface with the 'bank_database' selected. In the SQL Editor tab, the following SQL code is written:

```

91 * bankAccount where depositor.acco = bankAccount.acco
92 * group by customer_name,branch_name
93 * having count(depositor.acco)>=2
94 *
95 * create view BranchLoanSummary as
96 * select branch_name,sum(amount) as total_loan_amount
97 * from loan group by branch_name;
98 *
99 * select * from BranchLoanSummary;

```

The Result Grid shows the output of the last query:

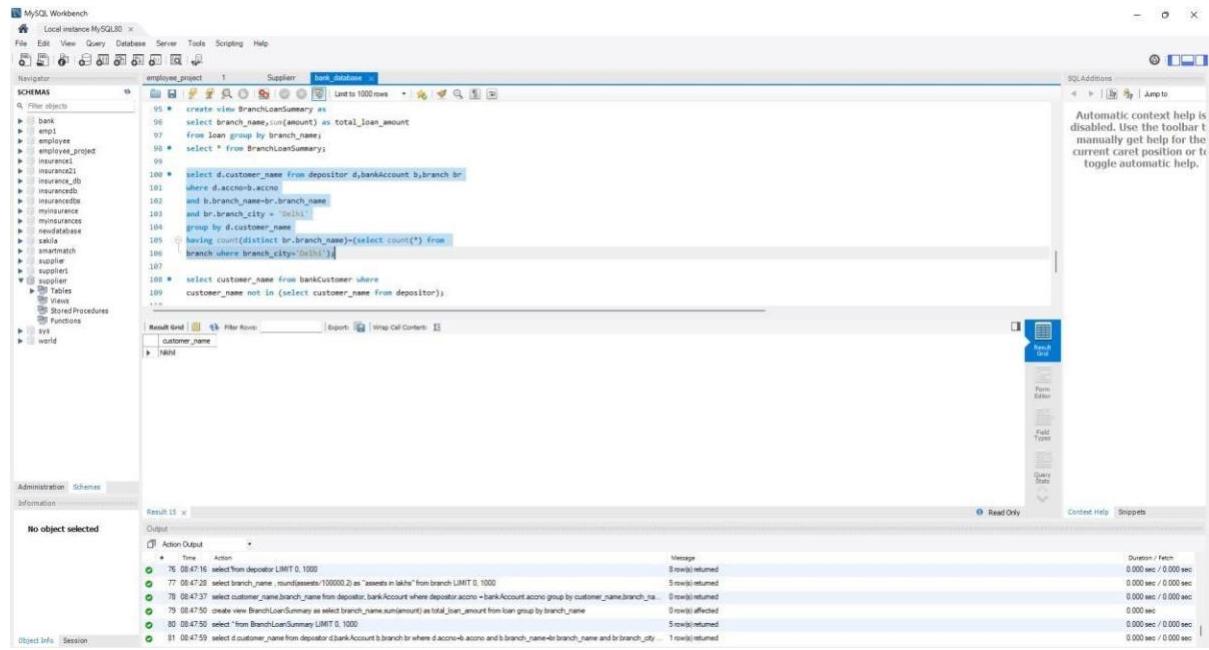
branch_name	total_loan_amount
SBI_ChennaiRoad	2000
SBI_GovtEmployeeRoad	5000
SBI_PerimeterRoad	4000
SBI_ResidencyRoad	2000
SBI_VineetRoad	3000

The Activity Log at the bottom shows the execution history:

Action	Time	Message	Duration / Fetch
75 08:47:04	selected from loan LIMIT 0, 1000	5 rows returned	0.000 sec / 0.000 sec
76 08:47:16	selected from depositor LIMIT 0, 1000	3 rows returned	0.000 sec / 0.000 sec
77 08:47:28	select branch_name , round(assets/100000,2) as "assets in lakhs" from branch LIMIT 0, 1000	5 rows returned	0.000 sec / 0.000 sec
78 08:47:37	select customer_name,branch_name from depositor, bankAccount where depositor.acco = bankAccount.acco group by customer_name,branch_name;	0 rows returned	0.000 sec / 0.000 sec
79 08:47:50	create view BranchLoanSummary as select branch_name,sum(amount) as total_loan_amount from loan group by branch_name;	0 rows affected	0.000 sec
80 08:47:50	select * from BranchLoanSummary LIMIT 0, 1000	5 rows returned	0.000 sec / 0.000 sec

Experiment 4: More Queries on Bank Database

Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi).



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

```
95 * create view BranchLoanSummary as
96 select branch_name,sum(amount) as total_loan_amount
97 from loan group by branch_name;
98 *
99 select *
100 from BranchLoanSummary;
```

Below the editor, the results grid displays a single row of data:

customer_name
Ram

The status bar at the bottom indicates "Result 1/1" and "Read Only".

Find all customers who have a loan at the bank but do not have an account.

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, Help. The left sidebar displays the 'Schemas' tree, which includes 'employee_project' (selected), 'Supplier', and 'bank'. The main area contains a large block of SQL code:

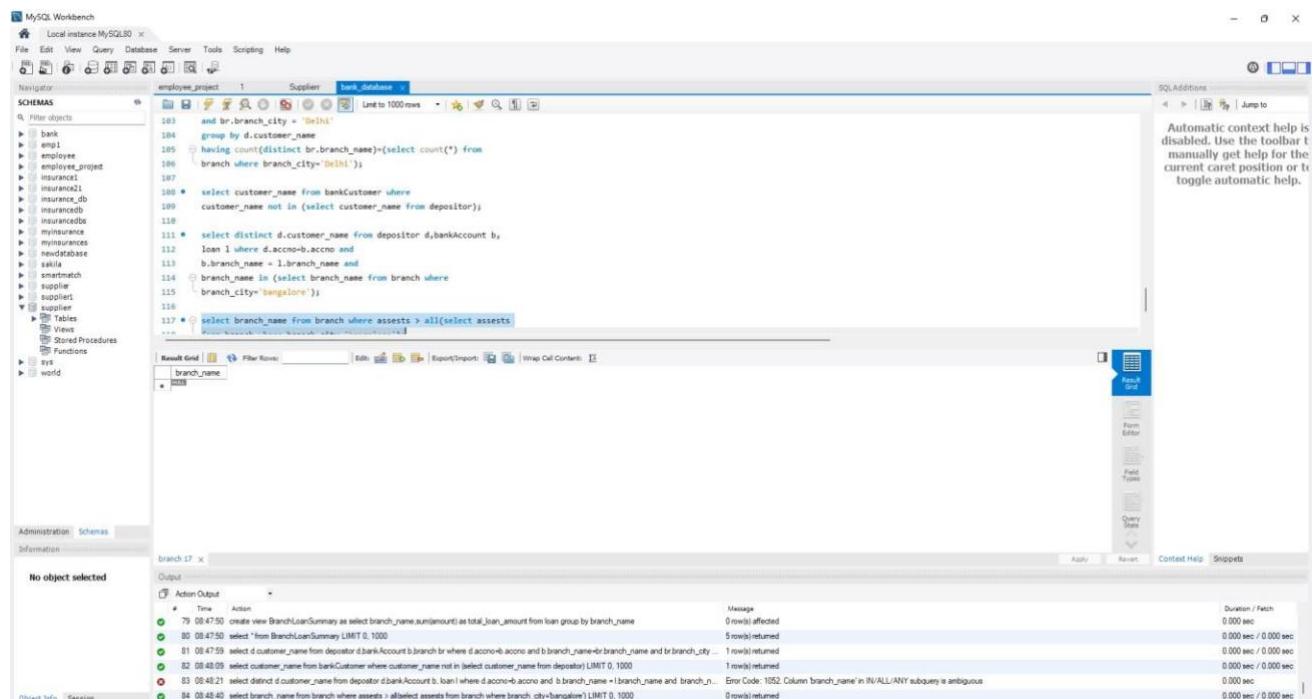
```
95 • create view branchLoanSummary as
96 select branch_name,sum(amount) as total_loan_amount
97 from loan group by branch_name;
98 • select * from BranchLoanSummary;
99
100 • select d.customer_name from depositor d,bankAccount b,branch br
101 where d.acno=b.acno
102 and b.branch_name=br.branch_name
103 and br.branch_city = 'Delhi'
104 group by d.customer_name
105 having count(distinct br.branch_name)=(select count(*) from
106 branch where branch_city='Delhi');
107
108 • select customer_name from bankCustomer where
109 customer_name not in (select customer_name from depositor);
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
```

Find all customers who have both an account and a loan at the Bangalore branch.

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** employee_project (selected), Supplier, bank_database.
- Navigator:** Schemas, Tables, Views, Stored Procedures, Functions, sys, world.
- SQL Editor:** Contains a multi-line SQL script for creating a view, selecting from multiple tables (depositor, bankAccount, branch, bankCustomer), and performing joins and subqueries to calculate total loan amounts by branch name. The code includes several comments and annotations.
- Output Tab:** Action Output, showing the execution results for each statement in the script.
- Help Bar:** SQLAdditions, Jump to, Automatic context help is disabled. Use the toolbar or manually get help for the current caret position or to toggle automatic help.

Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).



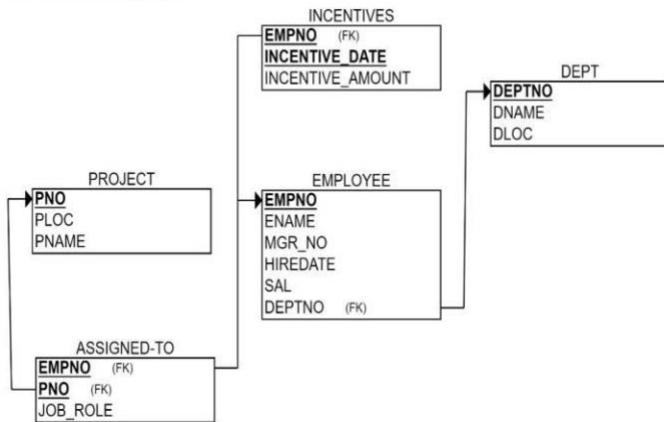
Experiment 5: Employee DATABASE

The employee database must record each employee's identifying number, name, manager reference, hire date, salary, and department affiliation while also tracking departmental details, project assignments (including the role an employee plays on a project), and any incentive payments given to employees. Every employee is represented by a unique employee number and has a hire date and salary that must be valid; the manager field is a self-referencing link that must, if present, point to an existing employee and must never create a circular management chain or reference the employee themselves. Departments are identified by a unique department number and include a department name and location; every department referenced by an employee or by other structures must exist in the department table, and departments may contain zero or many employees. Projects are recorded with a unique project number, project name and project location; employees may be assigned to multiple projects and each project may have many employees, with each assignment carrying the employee's job role for that project — duplicate assignments of the same employee to the same project are disallowed. Incentive payments are recorded with the employee reference, the incentive date and the incentive amount; an incentive entry must reference an existing employee and incentive amounts must be non-negative and dated on or after the employee's hire date. Referential integrity must be enforced so that employee records cannot reference non-existent departments, projects, or managers, and assignment and incentive records cannot exist without corresponding employee, project, or department records as appropriate. Salary, incentive amounts, and any monetary fields must be constrained to valid numeric ranges and hire/ incentive dates must be valid calendar dates (and typically not future-dated unless business rules permit). Deletion and update policies must preserve historical consistency: deleting an employee who appears as a manager, as a project assignee, or in incentive records should be prevented or should be handled via controlled archival, reassignment, or soft-delete flags rather than hard deletion to preserve audit trails; similarly, changing a department or project identifier must either be disallowed if it would orphan historical records or handled by introducing immutable surrogate keys. Business rules include preventing circular manager chains, ensuring an employee's manager (if specified) cannot be the employee themselves, disallowing duplicate project-assignments, requiring that incentive dates fall within the employee's employment window, and optionally requiring at least one project assignment or at least one incentive record depending on policy for reporting. Implementation should use primary-key and foreign-key constraints for identity and linkage, unique constraints to prevent duplicate assignments, check constraints for monetary and date ranges, and application logic or triggers for complex temporal or graph constraints (like cycle detection in management relationships and enforcing non-overlap or other schedule-related rules if assignments gain temporal attributes later). The system must therefore reliably support queries such as employee reporting lines, department staffing lists, project rosters

with job roles, incentive payment histories, salary analyses, and audit reports while maintaining data integrity, preventing inconsistent deletions, and preserving a complete historical record for HR and compliance needs.

Schema Diagram

Schema Diagram



Create Database

```
create database Employee_project; show  
databases;  
use employee_project;
```

Create table

```
CREATE TABLE DEPT (  
DEPTNO INT PRIMARY KEY,  
DNAME VARCHAR(30) NOT NULL,  
DLOC VARCHAR(30) NOT NULL  
);  
desc dept;  
  
CREATE TABLE EMPLOYEE (  
EMPNO INT PRIMARY KEY,  
ENAME VARCHAR(30) NOT NULL,  
MGR_NO INT,  
HIREDATE DATE,  
SAL DECIMAL(10,2),  
DEPTNO INT,  
FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO)  
);  
desc employee;
```

```

CREATE TABLE PROJECT (
    PNO INT PRIMARY KEY,
    PLOC VARCHAR(30) NOT NULL,
    PNAME VARCHAR(30) NOT NULL
);
desc project;

CREATE TABLE ASSIGNED_TO (
    EMPNO INT,
    PNO INT,
    JOB_ROLE VARCHAR(30),
    PRIMARY KEY (EMPNO, PNO),
    FOREIGN KEY (EMPNO) REFERENCES EMPLOYEE(EMPNO),
    FOREIGN KEY (PNO) REFERENCES PROJECT(PNO)
);
desc assigned_to;

CREATE TABLE INCENTIVES (
    EMPNO INT,
    INCENTIVE_DATE DATE,
    INCENTIVE_AMOUNT DECIMAL(10,2),
    PRIMARY KEY (EMPNO, INCENTIVE_DATE),
    FOREIGN KEY (EMPNO) REFERENCES EMPLOYEE(EMPNO) );

```

Structure of the table desc

dept;

	Field	Type	Null	Key	Default	Extra
▶	deptno	int	NO	PRI	NULL	
	dname	varchar(30)	YES		NULL	
	dloc	varchar(30)	YES		NULL	

desc employee;

	Field	Type	Null	Key	Default	Extra
▶	empno	int	NO	PRI	NULL	
	ename	varchar(30)	YES		NULL	
	mgr_no	int	YES		NULL	
	hiredate	date	YES		NULL	
	sal	decimal(10,2)	YES		NULL	
	deptno	int	YES	MUL	NULL	

desc project;

	Field	Type	Null	Key	Default	Extra
▶	pno	int	NO	PRI	NULL	
	pname	varchar(30)	YES		NULL	
	dloc	varchar(30)	YES		NULL	

desc assigned;

	Field	Type	Null	Key	Default	Extra
▶	empno	int	NO	PRI	NULL	
	pno	int	NO	PRI	NULL	
	job_role	varchar(30)	YES		NULL	

desc incentives;

	Field	Type	Null	Key	Default	Extra
▶	empno	int	YES	MUL	NULL	
	incentives_date	date	YES		NULL	
	incentives_amount	decimal(10,2)	YES		NULL	

Insertion of values into Table

```
INSERT INTO DEPT VALUES  
(10, 'Sales', 'Bengaluru'),  
(20, 'Accounting', 'Hyderabad'),  
(30, 'Research', 'Mysuru'),  
(40, 'Operations', 'Chennai'),  
(50, 'HR', 'Mumbai'),  
(60, 'IT', 'Delhi');  
select * from dept;
```

employee_project* 1

```

49
50 • INSERT INTO DEPT VALUES
51   (10, 'Sales', 'Bengaluru'),
52   (20, 'Accounting', 'Hyderabad'),
53   (30, 'Research', 'Mysuru'),
54   (40, 'Operations', 'Chennai'),
55   (50, 'HR', 'Mumbai'),
56   (60, 'IT', 'Delhi')
57 • select * from dept
58

59 • INSERT INTO EMPLOYEE VALUES
60   (1001, 'Alice', NULL, '2019-02-10', 75000, 10),
61   (1002, 'Bob', 1001, '2020-06-01', 65000, 20),
62   (1003, 'Charlie', 1001, '2018-09-15', 80000, 30),
63   (1004, 'Diana', 1002, '2021-01-20', 55000, 40),
64   (1005, 'Ethan', 1003, '2022-07-12', 60000, 50),
65   (1006, 'Fay', 1001, '2023-03-05', 52000, 10);
66 • select * from employee;
67

68 • INSERT INTO PROJECT VALUES

```

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

DEPTNO	DNAME	DLLOC
10	Sales	Bengaluru
20	Accounting	Hyderabad
30	Research	Mysuru
40	Operations	Chennai
50	HR	Mumbai
60	IT	Delhi

Result Grid Form Editor

INSERT INTO EMPLOYEE VALUES

```
(1001, 'Alice', NULL, '2019-02-10', 75000, 10),
(1002, 'Bob', 1001, '2020-06-01', 65000, 20),
(1003, 'Charlie', 1001, '2018-09-15', 80000, 30),
(1004, 'Diana', 1002, '2021-01-20', 55000, 40),
(1005, 'Ethan', 1003, '2022-07-12', 60000, 50),
(1006, 'Fay', 1001, '2023-03-05', 52000, 10);
```

```
select * from employee;
```

employee_project* 1

```

58
59 • INSERT INTO EMPLOYEE VALUES
60   (1001, 'Alice', NULL, '2019-02-10', 75000, 10),
61   (1002, 'Bob', 1001, '2020-06-01', 65000, 20),
62   (1003, 'Charlie', 1001, '2018-09-15', 80000, 30),
63   (1004, 'Diana', 1002, '2021-01-20', 55000, 40),
64   (1005, 'Ethan', 1003, '2022-07-12', 60000, 50),
65   (1006, 'Fay', 1001, '2023-03-05', 52000, 10);
66 • select * from employee;
67

68 • INSERT INTO PROJECT VALUES
69   (200, 'Bengaluru', 'Alpha'),
70   (201, 'Hyderabad', 'Beta'),
71   (202, 'Mysuru', 'Gamma'),
72   (203, 'Chennai', 'Delta'),
73   (204, 'Mumbai', 'Epsilon'),
74   (205, 'Pune', 'Zeta');
75 • select * from project;
76

77 • INSERT INTO ASSIGNED_TO VALUES

```

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

EMPHO	ENAME	MGR_NO	HIREDATE	SAL	DEPTNO
1001	Alice	NULL	2019-02-10	75000.00	10
1002	Bob	1001	2020-06-01	65000.00	20
1003	Charlie	1001	2018-09-15	80000.00	30
1004	Diana	1002	2021-01-20	55000.00	40
1005	Ethan	1003	2022-07-12	60000.00	50
1006	Fay	1001	2023-03-05	52000.00	10

Result Grid Form Editor

INSERT INTO PROJECT VALUES

```
(200, 'Bengaluru', 'Alpha'),  
(201, 'Hyderabad', 'Beta'),  
(202, 'Mysuru', 'Gamma'),  
(203, 'Chennai', 'Delta'),  
(204, 'Mumbai',  
'Epsilon'), (205, 'Pune',  
'Zeta'); select * from  
project;
```

The screenshot shows the Oracle SQL Developer interface with the following details:

- SQL Editor:** Contains the following SQL code:

```
employee_project < 1  
64  (1005, 'Ethan', 1003, '2022-07-12', 60000, 50),  
65  (1006, 'Fay', 1001, '2023-03-05', 52000, 10);  
66 • select * from employee;  
67  
68 • INSERT INTO PROJECT VALUES  
69  (200, 'Bengaluru', 'Alpha'),  
70  (201, 'Hyderabad', 'Beta'),  
71  (202, 'Mysuru', 'Gamma'),  
72  (203, 'Chennai', 'Delta'),  
73  (204, 'Mumbai', 'Epsilon'),  
74  (205, 'Pune', 'Zeta');  
75 • select * from project;  
76  
77 • INSERT INTO ASSIGNED_TO VALUES  
78  (1001, 200, 'Lead'),  
79  (1002, 201, 'Analyst'),  
80  (1003, 202, 'Senior'),  
81  (1004, 203, 'Support'),  
82  (1005, 204, 'Recruiter'),  
83  (1006, 200, 'Developer'),
```
- Result Grid:** Displays the results of the last query:

PNO	PLOC	PNAME
200	Bengaluru	Alpha
201	Hyderabad	Beta
202	Mysuru	Gamma
203	Chennai	Delta
204	Mumbai	Epsilon
205	Pune	Zeta
1005	1003	1003

INSERT INTO ASSIGNED_TO VALUES

```
(1001, 200, 'Lead'),  
(1002, 201, 'Analyst'),  
(1003, 202, 'Senior'),  
(1004, 203, 'Support'),  
(1005, 204, 'Recruiter'),  
(1006, 200, 'Developer'),  
(1002, 203, 'Tester'),  
(1003, 200, 'Consultant');  
select * from assigned_to;
```

```
employee_project x 1
Limit to 1000 rows
73   (284, 'Mumbai', 'Epsilon'),
74   (285, 'Pune', 'Zeta');
75 • select * from projects;
76
77 • INSERT INTO ASSIGNED_TO VALUES
78   (1001, 200, 'Lead'),
79   (1002, 201, 'Analyst'),
80   (1003, 202, 'Senior'),
81   (1004, 203, 'Support'),
82   (1005, 204, 'Recruiter'),
83   (1006, 200, 'Developer'),
84   (1002, 203, 'Tester'),
85   (1003, 200, 'Consultant');
86 • select * from assigned_to;
87
88 • INSERT INTO INCENTIVES VALUES
89   (1001, '2024-01-15', 1500.00),
90   (1002, '2024-02-10', 1200.00),
91   (1003, '2024-03-05', 1800.00),
92   (1005, '2024-05-20', 1000.00),
```

INSERT INTO INCENTIVES VALUES

```
(1001, '2024-01-15', 1500.00),  
(1002, '2024-02-10', 1200.00),  
(1003, '2024-03-05', 1800.00),  
(1005, '2024-05-20', 1000.00),  
(1006, '2024-06-18', 900.00),  
(1001, '2024-07-01', 500.00);  
select * from incentives;
```

employee_project

```
Limit to 1000 rows
```

85 • `(1003, 200, 'Consultant');`

86 • `select * from assigned_to;`

87

88 • `INSERT INTO INCENTIVES VALUES`

89 `(1001, '2024-01-15', 1500.00),`

90 `(1002, '2024-02-10', 1200.00),`

91 `(1003, '2024-03-05', 1800.00),`

92 `(1005, '2024-05-20', 1000.00),`

93 `(1006, '2024-06-18', 900.00),`

94 `(1001, '2024-07-01', 500.00);`

95 • `select * from incentives;`

96

97 • `SELECT DISTINCT a.EMPNO`

98 `FROM ASSIGNED_TO a`

99 `JOIN PROJECT p ON a.PNO = p.PNO`

100 `WHERE p.PLOC IN ('Bengaluru', 'Hyderabad', 'Mysuru');`

101

102 • `SELECT e.EMPNO`

103 `FROM EMPLOYEE e`

104 `LEFT JOIN INCENTIVES i ON e.EMPNO = i.EMPNO`

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

EMPNO	INCENTIVE_DATE	INCENTIVE_AMOUNT
1001	2024-01-15	1500.00
1001	2024-07-01	500.00
1002	2024-02-10	1200.00
1003	2024-03-05	1800.00
1005	2024-05-20	1000.00
1006	2024-06-18	900.00

incentives 32 x

Queries

Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru.

The screenshot shows the Oracle SQL Developer interface with a query editor window titled "employee_project" containing the following SQL code:

```
86 • select * from assigned_to;
87
88 • INSERT INTO INCENTIVES VALUES
89   (1001, '2024-01-15', 1500.00),
90   (1002, '2024-02-10', 1200.00),
91   (1003, '2024-03-05', 1800.00),
92   (1005, '2024-05-20', 1000.00),
93   (1006, '2024-06-18', 900.00),
94   (1001, '2024-07-01', 500.00);
95 • select * from incentives;
96
97 • SELECT DISTINCT a.EMPNO
98   FROM ASSIGNED_TO a
99   JOIN PROJECT p ON a.PNO = p.PNO
100 WHERE p.PLOC IN ('Bengaluru', 'Hyderabad', 'Mysuru');
101
102 • SELECT e.EMPNO
103   FROM EMPLOYEE e
104   LEFT JOIN INCENTIVES i ON e.EMPNO = i.EMPNO
105 WHERE i.EMPNO IS NULL;
```

The result grid shows the following data:

EMPNO
1001
1003
1006
1002

Get Employee ID's of those employees who didn't receive incentives

The screenshot shows the Oracle SQL Developer interface with a query editor window titled "employee_project" containing the following SQL code:

```
86 • select * from assigned_to;
87
88 • INSERT INTO INCENTIVES VALUES
89   (1001, '2024-01-15', 1500.00),
90   (1002, '2024-02-10', 1200.00),
91   (1003, '2024-03-05', 1800.00),
92   (1005, '2024-05-20', 1000.00),
93   (1006, '2024-06-18', 900.00),
94   (1001, '2024-07-01', 500.00);
95 • select * from incentives;
96
97 • SELECT DISTINCT a.EMPNO
98   FROM ASSIGNED_TO a
99   JOIN PROJECT p ON a.PNO = p.PNO
100 WHERE p.PLOC IN ('Bengaluru', 'Hyderabad', 'Mysuru');
101
102 • SELECT e.EMPNO
103   FROM EMPLOYEE e
104   LEFT JOIN INCENTIVES i ON e.EMPNO = i.EMPNO
105 WHERE i.EMPNO IS NULL;
```

The result grid shows the following data:

EMPNO
1004

Write a SQL query to find the employees name, number, dept, job_role, department location and project location who are working for a project location same as his/her department location.

The screenshot shows a MySQL Workbench window titled "employee_project". The SQL editor contains the following query:

```

94  (1001, '2024-07-01', 500.00);
95 *
96
97 *   SELECT DISTINCT a.EMPNO
98   FROM ASSIGNED_TO a
99   JOIN PROJECT p ON a.PNO = p.PNO
100  WHERE p.PLOC IN ('Bengaluru','Hyderabad','Mysuru');
101
102 *   SELECT e.ENAME, e.EMPNO, e.DEPTNO, a.JOB_ROLE,
103     d.DLOC AS DEPT_LOCATION, p.PLOC AS PROJECT_LOCATION
104   FROM EMPLOYEE e
105   JOIN DEPT d ON e.DEPTNO = d.DEPTNO
106   JOIN ASSIGNED_TO a ON e.EMPNO = a.EMPNO
107   JOIN PROJECT p ON a.PNO = p.PNO
108   WHERE d.DLOC = p.PLOC;
109
110
111
112
113

```

The results grid displays the following data:

	ENAME	EMPNO	DEPTNO	JOB_ROLE	DEPT_LOCATION	PROJECT_LOCATION
▶	Alice	1001	10	Lead	Bengaluru	Bengaluru
	Bob	1002	20	Analyst	Hyderabad	Hyderabad
	Charlie	1003	30	Senior	Mysuru	Mysuru
	Diana	1004	40	Support	Chennai	Chennai
	Ethan	1005	50	Recruiter	Mumbai	Mumbai
	Fay	1006	10	Developer	Bengaluru	Bengaluru

Experiment 6: More Queries on Employee Database

List the name of the managers with the maximum employees.

The screenshot shows the MySQL Workbench interface with the following details:

Schemas: Local instance MySQL80, employee_project

Query Editor:

```
100 FROM Employee e
101 JOIN DEPT d ON e.DEPTNO = d.DEPTNO
102 JOIN ASSIGNED_TO a ON e.EMPNO = a.EMPNO
103 JOIN PROJECT p ON a.PNO = p.PNO
104 WHERE d.DLOC = p.DLOC;
105
106 * select e.Ename as manager_name, count(e.EmpNo) as Num_Employees
107 from Employee e
108 Join Employee m on e.EmpNo = m.Mgr_No
109 Group by e.Ename
110 having count(e.EmpNo)=(
111     select max(cnt)
112     from (
113         select count(Mgr_No) as cnt
114         from Employee
115         where Mgr_No is not null
116         group by Mgr_No
117     ) as temp
118 );
```

Result Grid:

manager_name	Num_Employees
Alice	3

Output:

Action	Time	Message	Duration / Fetch
show databases	2 08:24:58	21 rows returned	0.000 sec / 0.000 sec
create database employee_project	3 08:24:58	Error Code: 1007. Can't create database 'employee_project'; database exists	0.000 sec
select e.Ename as manager_name, countin(EmpNo) as Num_Employees from Employee e Join Employee m on e.EmpNo = m.Mgr_No Group by e.Ename;	4 08:25:18	Error Code: 1046. No database selected. Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.	0.000 sec
create database employee_project	5 08:25:43	Error Code: 1007. Can't create database 'employee_project'; database exists	0.000 sec
use employee_project	6 08:25:43	0 rows affected	0.000 sec
select e.Ename as manager_name, countin(EmpNo) as Num_Employees from Employee e Join Employee m on e.EmpNo = m.Mgr_No Group by e.Ename;	7 08:25:50	1 row(s) returned	0.000 sec / 0.000 sec

Display those managers name whose salary is more than average salary of his employee.

The screenshot shows the MySQL Workbench interface with the following details:

Schemas: Local instance MySQL80, employee_project

Query Editor:

```
141 from employee e
142 join dept d on e.deptno = d.deptno
143 where e.mgr_no in (
144 select empno from employee where mgr_no is null
145 );
146
147 * select empno,incentive_date,incentive_amount
148 from incentives
149 where year(incentive_date)> 2010 and month(incentive_date)=1
150 order by incentive_amount desc
151 limit 1 offset 1;
152
153 * select e.ename as employee_name,e.ename as manager_name ,
154 d.dname as department
155 from employee e
156 join employee m on e.mgr_no = m.empno
157 join dept d on e.deptno = d.deptno
158 where e.deptno = m.deptno;
```

Result Grid:

employee_name	manager_name	department
Pay	Alice	Sales

Output:

Action	Time	Message	Duration / Fetch
select e.ename as Manager_Name from Employee e where e.EmpNo in (select distinct Mgr_No from Employee where Mgr_No is not null) and e.sal <...	9 08:30:17	3 rows(s) returned	0.000 sec / 0.000 sec
select e.ename as Manager_Name from Employee e where e.EmpNo in (select distinct Mgr_No from Employee where Mgr_No is not null) and e.sal <...	10 08:30:41	3 rows(s) returned	0.000 sec / 0.000 sec
select e.ename as second_level_manager,d.dname as department	11 08:33:10	3 rows(s) returned	0.000 sec / 0.000 sec
from employee e join dept d on e.deptno = d.deptno where e.mgr_no in (select en...	12 08:36:01	0 rows(s) returned	0.000 sec / 0.000 sec
select empno,incentive_date,incentive_amount from incentives where year(incentive_date)> 2010 and month(incentive_date)=1 order by incentive_am...	13 08:44:05	Error Code: 1054. Unknown column 'incentive_name' in 'field list'	0.000 sec
select e.ename as employee_name,m.ename as manager_name ,d.dname as department from employee e join employee m on e.mgr_no = m.empno jo...	14 08:44:22	1 rows(s) returned	0.000 sec / 0.000 sec

Find the employee details who got second maximum incentive in January 2019.

The screenshot shows the MySQL Workbench interface. The top navigation bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar displays the Navigator and Schemas sections, with 'employee_project' selected. The main area contains a query editor with the following SQL script:

```
1 employee_project
2
3 131 * select e.Ename as Manager_Name
4 132   from Employee e
5 133   where e.EmpNo in (select distinct Mgr_No from Employee where Mgr_No is not null)
6 134   and e.sal > (
7 135     select avg(s.sal)
8 136     from employee s
9 137     where s.Mgr_no = e.emplo
10 138   )
11
12 139
13 140 * select e.ename as second_level_manager,d.dname as department
14 141   from employee e
15 142   join dept d on e.deptno = d.deptno
16 143   where e.mgr_no in (
17 144     select empno from employee where mgr_no is null
18 145   )
19
20 146
21 147 * select empno,incentive_date,incentive_amount
22 148   from incentives
23 149   where year(incentive_date)> 2010 and month(incentive_date)=1
24 150   order by incentive_amount desc
```

The results grid below the script shows the output for the first query:

second_level_manager	department
Ray	Sales
Bob	Accounting
Charlie	Research

The bottom right corner features a timeline panel titled 'SQLAdditions' with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.' Below the timeline are buttons for 'Result Grid', 'Filter Rows', 'Import', 'Wrap Cell Contents', 'Read Only', 'Contact Help', and 'Snippets'. The status bar at the bottom right indicates 'Activate Windows' and provides links to 'Go to Settings to activate View' and '0.000 sec / 0.000 sec'.

Display those employees who are working in the same department where his manager is working

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** employee_project
- Code Area:** A large text area containing a complex SQL query. The code is as follows:

```
119 Join Employee m on e.EmpNo = m.Mgr_No
120 Group by e.Dname
121 having count(e.EmpNo) >=
122 select max(cnt)
123 from (
124 select count(Mgr_No) as cnt
125 From Employee
126 where Mgr_No is not null
127 Group by Mgr_No
128 ) as temp
129 )
130
131 select e.Dname as Manager_Name
132 from Employee e
133 where e.EmpNo in (select distinct Mgr_No from Employee where Mgr_No is not null)
134 and e.Sal > (
135 select avg(s.Sal)
136 From employee s
137 where s.Mgr_no = e.EmpNo
138 )
```

- Result Grid:** A table showing results for Manager_Name. The data is:

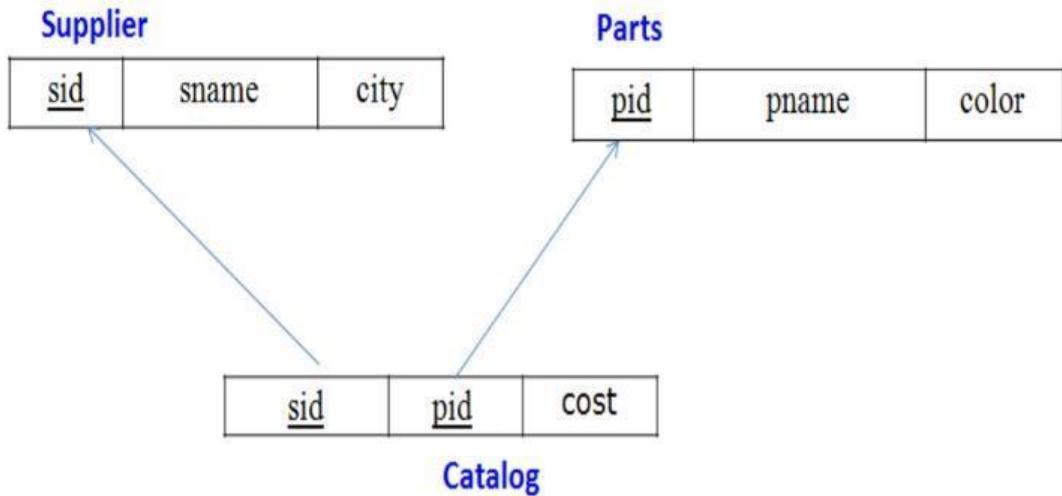
Manager_Name
Alice
Bob
Charlie

- Log:** Shows the execution history of the query.
- Toolbar:** Includes icons for Run, Stop, Refresh, and other database management functions.
- Help:** A message in the top right corner: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Experiment no7 Supplier Database

The supplier database must store information about suppliers, the parts they provide, and the prices at which each part is offered so that purchasing, analysis, and reporting can be done accurately. Each supplier is uniquely identified by a supplier ID and is recorded with a name and the city in which the supplier is located; each part is uniquely identified by a part ID and includes a part name and a color. The system must maintain a catalog that links suppliers to the parts they supply and records the cost at which a given supplier sells a given part. Every catalog entry must reference an existing supplier and an existing part, and there must be no duplicate entries for the same combination of supplier and part, so that at most one current price record exists per supplier–part pair. Costs must be valid numeric values and strictly non-negative, and business rules may specify upper limits or currency formats that must be enforced consistently. The data model must support the possibility that a supplier can provide many different parts, that a part can be supplied by many different suppliers, and that some suppliers or parts may temporarily have no catalog entries if they are inactive or not currently traded. Referential integrity must be enforced so that a supplier or part cannot be deleted while still referenced in the catalog unless such deletion is handled by controlled archival or cascade rules that preserve historical price information; in general, historical catalog data should not be lost, as it may be required for audits or trend analysis. The system should allow queries such as “find all suppliers for a given part,” “list all parts provided by a given supplier,” “retrieve the cheapest supplier for each part,” and “analyze supplier coverage by city,” and must therefore guarantee that identifiers are unique, relationships between suppliers, parts, and catalog entries are consistent, and price information is accurate and reliably maintained over time.

Schema Diagram



Create database

```
create database if not exists Supplierr ; use  
supplierr;
```

Create Table

```
CREATE TABLE Supplier (  
    sid INT PRIMARY KEY,  
    sname VARCHAR(50),  
    city  
    VARCHAR(50)  
) ;
```

```
CREATE TABLE Parts (  
    pid INT PRIMARY KEY,  
    pname VARCHAR(50),  
    color VARCHAR(20)  
) ;
```

```
CREATE TABLE Catalog (
```

```

    sid INT,
    pid INT,
    cost INT,
    PRIMARY KEY (sid, pid),
    FOREIGN KEY (sid) REFERENCES Supplier(sid),
    FOREIGN KEY (pid) REFERENCES Parts(pid)
);

```

Structure of the table

```
desc supplier;
```

	Field	Type	Null	Key	Default	Extra
▶	sid	int	NO	PRI	NULL	
	sname	varchar(40)	YES		NULL	
	city	varchar(40)	YES		NULL	

```
desc parts;
```

	Field	Type	Null	Key	Default	Extra
▶	pid	int	NO	PRI	NULL	
	pname	varchar(40)	YES		NULL	
	colour	varchar(20)	YES		NULL	

```
desc catalog;
```

	Field	Type	Null	Key	Default	Extra
▶	sid	int	YES	MUL	NULL	
	pid	int	YES	MUL	NULL	
	cost	int	NO	PRI	NULL	

Insertion of values into the table

```
INSERT INTO Supplier VALUES  
(10001, 'Acme Widget', 'Bangalore'),  
(10002, 'Johns', 'Kolkata'),  
(10003, 'Vimal', 'Mumbai'),  
(10004, 'Reliance', 'Delhi');
```

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** employee_project
- Tables:** Supplier, Parts, Catalog
- SQL Editor:** Contains the following SQL code:

```
20 * )  
21 * desc catalogs;  
22 *  
23 * INSERT INTO Supplier VALUES  
24 * (10001, "Acme widget", "Bangalore");  
25 * (10002, "Johns", "Kolkata");  
26 * (10003, "Vimal", "Mumbai");  
27 * (10004, "Reliance", "Delhi");  
28 * select * from supplier;  
29 *  
30 * INSERT INTO Parts VALUES  
31 * (20001, "Book", "Red");  
32 * (20002, "Pen", "Red");  
33 * (20003, "Pencil", "Green");  
34 * (20004, "Mobile", "Green");  
35 * (20005, "Charger", "Black");  
36 * select * from parts;  
37 *  
38 *  
39 * INSERT INTO Catalog VALUES  
40 * (30001, 20001, 10);
```

- Result Grid:** Displays the inserted data for the Supplier table.

id	name	city
10001	Acme Widget	Bangalore
10002	Johns	Kolkata
10003	Vimal	Mumbai
10004	Reliance	Delhi

- Output:** Shows the execution log with the following entries:

Action	Time	Message	Duration / Fetch
use supplier	08:28:40	0 row(s) affected	0.000 sec
desc supplier	08:28:44	3 row(s) returned	0.000 sec / 0.000 sec
desc parts	08:29:10	3 row(s) returned	0.000 sec / 0.000 sec
desc catalog	08:29:24	3 row(s) returned	0.000 sec / 0.000 sec
select * from supplier LIMIT 0, 1000	08:29:38	4 row(s) returned	0.016 sec / 0.000 sec
select * from supplier LIMIT 0, 1000	08:29:41	4 row(s) returned	0.000 sec / 0.000 sec

```
INSERT INTO Parts VALUES  
(20001, 'Book', 'Red'),  
(20002, 'Pen', 'Red'),  
(20003, 'Pencil', 'Green'),  
(20004, 'Mobile', 'Green'),  
(20005, 'Charger', 'Black');
```

```

MySQL Workbench - Local instance MySQL> <
File Edit View Query Database Server Tools Scripting Help
SQL File 5 | Supplier | employee_project | Limit to 1000 rows | SQLAdditions | Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Navigator Schemas employee_project Supplier SQL File 5
Schemas
employee_project
bank
emp1
employee
insurance
insurance1
insurance2
insurance_db
insurance_m
insurancebs
myinsurance
myinsurances
smartschema
smartschema
supplier
Tables Views Stored Procedures
sys Functions
world

Results Grid | Filter Rows | Edit | Export/Import | Wrap Cell Content | Apply | Context Help | Snippets
parts | pd | pname | color |
20001 Book Red
20002 Pen Red
20003 Grid Green
20004 Mobile Green
20005 Charger Black
20006 Headphones Black
20007 Headphones Black
20008 Headphones Black
20009 Headphones Black
20010 Headphones Black

Administration Schemas Information
No object selected
parts # x
Output
Action Output
# 2 00:28:44 Action
2 00:28:44 desc supplier
3 00:29:10 desc parts
4 00:29:24 desc catalog
5 00:29:38 select * from supplier LIMIT 0, 1000
6 00:29:41 select * from supplier LIMIT 0, 1000
7 00:29:53 select * from parts LIMIT 0, 1000

```

INSERT INTO Catalog VALUES
(10001, 20001, 10),
(10001, 20002, 10),
(10001, 20003, 30),
(10001, 20004, 10),
(10002, 20001, 10),
(10002, 20002, 10),
(10003, 20003, 30),
(10004, 20003, 40);

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Schemas

employee_project

SQL File 5

```

37  (20001, 'Pen', 'Red'),
38  (20001, 'Pencil', 'Green'),
39  (20001, 'Mobile', 'Green'),
40  (20001, 'Charger', 'Black')
41 * select * from parts
42 *
43 * INSERT INTO Catalog VALUES
44  (10001, 20001, 10),
45  (10001, 20002, 10),
46  (10001, 20003, 30),
47  (10001, 20004, 10),
48  (10002, 20001, 10),
49  (10002, 20002, 10),
50  (10002, 20003, 30),
51  (10002, 20004, 40)
52 * select * from catalog
53 *
54 * SELECT DISTINCT p.product_name
55 FROM Parts p
56 JOIN Catalog c ON p.part_id = c.part_id

```

Result Grid

part_id	product_id	cost
10001	20001	10
10001	20002	10
10001	20003	30
10001	20004	10
10002	20001	10
10002	20002	10
10002	20003	30
10002	20004	40
10003	20001	10
10003	20002	10
10003	20003	30
10004	20003	40
10004	20004	10

catalog 7

No object selected

Action Output

Action	Time	Message	Duration / Fetch
3	08:29:10	desc parts	0.000 sec / 0.000 sec
4	08:29:24	desc catalog	0.000 sec / 0.000 sec
5	08:29:38	select * from supplier LIMIT 0, 1000	0.016 sec / 0.000 sec
6	08:29:41	select * from supplier LIMIT 0, 1000	0.000 sec / 0.000 sec
7	08:29:53	select * from parts LIMIT 0, 1000	0.015 sec / 0.000 sec
8	08:30:10	select * from catalog LIMIT 0, 1000	0.000 sec / 0.000 sec

Queries

Find the pnames of parts for which there is some supplier

The screenshot shows the MySQL Workbench interface with the following details:

Query Editor:

```
SELECT DISTINCT p.pname
FROM Parts p
JOIN Catalog c ON p.pid = c.pid
JOIN Supplier s
ON s.sid = c.cid
WHERE NOT EXISTS (
    SELECT p.pid
    FROM Parts p
    WHERE p.pid NOT IN (
        SELECT c.pid
        FROM Catalog c
    )
)
```

Result Grid:

pname
Book
Pen
Pencil
Tablet

Output Window:

Time	Action	Message	Duration / Fetch
4 08:29:24	desc catalog	3 rows(s) returned	0.000 sec / 0.000 sec
5 08:29:30	select * from supplier LIMIT 0, 1000	4 rows(s) returned	0.016 sec / 0.000 sec
6 08:29:41	select * from supplier LIMIT 0, 1000	4 rows(s) returned	0.000 sec / 0.000 sec
7 08:29:53	select * from parts LIMIT 0, 1000	5 rows(s) returned	0.019 sec / 0.000 sec
8 08:30:10	select * from catalog LIMIT 0, 1000	8 rows(s) returned	0.000 sec / 0.000 sec
9 08:30:20	SELECT DISTINCT p.pname FROM Parts p JOIN Catalog c ON p.pid = c.pid LIMIT 0, 1000	4 rows(s) returned	0.000 sec / 0.000 sec

Find the snames of suppliers who supply every part

The screenshot shows the MySQL Workbench interface with the following details:

Query Editor:

```
SELECT s.sname
FROM Supplier s
WHERE NOT EXISTS (
    SELECT p.pid
    FROM Parts p
    WHERE p.color = 'Red'
    AND p.pid NOT IN (
        SELECT c.pid
        FROM Catalog c
        WHERE c.cid = s.sid
    )
)
```

Result Grid:

sname
Acme Widget

Output Window:

Time	Action	Message	Duration / Fetch
15 08:33:20	SELECT s.sname FROM Supplier s WHERE NOT EXISTS (SELECT p.pid FROM Parts p WHERE p.pid NOT IN (SELECT c.pid FROM Catalog c WHERE c.cid = s.sid))	1 rows(s) returned	0.000 sec / 0.000 sec
16 08:34:53	SELECT p.sname FROM Supplier s WHERE NOT EXISTS (SELECT p.pid FROM Parts p WHERE p.color = 'Red' AND p.pid NOT IN (SELECT c.pid FROM Catalog c WHERE c.cid = s.sid))	2 rows(s) returned	0.000 sec / 0.000 sec
17 08:34:59	SELECT p.sname FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.ad = c.ad WHERE s.sname = 'Acme Widget' AND p.pid NOT IN (SELECT c.pid FROM Catalog c WHERE c.cid = s.sid)	1 rows(s) returned	0.000 sec / 0.000 sec
18 08:35:20	SELECT DISTINCT c1.pid FROM Catalog c1 WHERE c1.cnt > (SELECT AVG(c2.cnt) FROM Catalog c2 WHERE c2.pid = p.pid LIMIT 0, 1)	1 rows(s) returned	0.000 sec / 0.000 sec
19 08:35:41	SELECT p.sname FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.ad = c.ad WHERE c.cnt > (SELECT MAX(c2.cnt) FROM Catalog c2 WHERE c2.pid = p.pid)	5 rows(s) returned	0.000 sec / 0.000 sec
20 08:37:20	SELECT s.sname FROM Supplier s WHERE NOT EXISTS (SELECT p.pid FROM Parts p WHERE p.pid NOT IN (SELECT c.pid FROM Catalog c WHERE c.cid = s.sid))	0 rows(s) returned	0.000 sec / 0.000 sec

Find the snames of suppliers who supply every red part.

Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.

Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- book
- empl
- employee
- employee_project
- insurance
- insurance1
- insurance_db
- insured
- myinsurance
- needatabase
- safile
- simpler
- supplier
- Supplier
- Tables
- Views
- Stored Procedures
- Functions
- sys
- world

Suppliers SQL File 5

```

85 JOIN Catalog c ON p.pid = c.pid
86 JOIN Supplier s ON s.sid = c.sid
87 WHERE s.name = 'Acme Widget'
88 AND p.pid NOT IN (
89     SELECT c2.pid
90     FROM Catalog c2
91     JOIN Supplier s2 ON c2.sid = s2.sid
92     WHERE s2.name < 'Acme Widget'
93 )
94
95 *   SELECT DISTINCT cl.sid
96   FROM Catalog cl
97 WHERE cl.cost > (
98     SELECT AVG(c2.cost)
99     FROM Catalog c2
100    WHERE c2.pid = cl.pid
101 )
102
103 *   SELECT p.pname, s.name
104   FROM Parts p

```

Result Grid

sd
10004

SQL Additions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas

No object selected

Action Output

Time	Action	Message	Duration / Fetch
18 08:39:25	SELECT DISTINCT cl.sid FROM Catalog cl WHERE cl.cost > (SELECT AVG(c2.cost)) FROM Catalog c2 WHERE c2.pid = cl.pid LIMIT 0... 1 rows returned		0.000 sec / 0.000 sec
19 08:35:41	SELECT p.pname, s.name FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.cid WHERE c.cost < (SELECT MAX(c2.co... 6 rows returned		0.000 sec / 0.000 sec
20 08:37:58	SELECT c.name FROM Supplier c WHERE NOT EXISTS (SELECT p FROM Parts p WHERE p.pid NOT IN (... 2 rows returned		0.000 sec / 0.000 sec
21 08:38:13	SELECT c.name FROM Supplier c WHERE NOT EXISTS (SELECT p FROM Parts p WHERE p.color = 'Red' AND p.pid NOT IN (... 2 rows returned		0.000 sec / 0.000 sec
22 08:38:31	SELECT p.pname FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.cid WHERE s.name = 'Acme Widget' AND p.pid NOT ... 1 rows returned		0.000 sec / 0.000 sec
23 08:38:48	SELECT DISTINCT cl.sid FROM Catalog cl WHERE cl.cost > (SELECT AVG(c2.cost)) FROM Catalog c2 WHERE c2.pid = cl.pid LIMIT 0... 1 rows returned		0.000 sec / 0.000 sec

Object Info Session

For each part, find the sname of the supplier who charges the most for that part.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- book
- empl
- employee
- employee_project
- insurance
- insurance1
- insurance_db
- insured
- myinsurance
- needatabase
- safile
- simpler
- supplier
- Supplier
- Tables
- Views
- Stored Procedures
- Functions
- sys
- world

Suppliers SQL File 5

```

94
95 *   SELECT DISTINCT cl.sid
96   FROM Catalog cl
97 WHERE cl.cost > (
98     SELECT AVG(c2.cost)
99     FROM Catalog c2
100    WHERE c2.pid = cl.pid
101 )
102
103 *   SELECT p.pname, s.name
104   FROM Parts p
105   JOIN Catalog c ON p.pid = c.pid
106   JOIN Supplier s ON s.sid = c.sid
107 WHERE c.cost = (
108     SELECT MAX(c2.cost)
109     FROM Catalog c2
110    WHERE c2.pid = p.pid
111 )
112
113

```

Result Grid

pname	sname
Book	Acme Widget
Pen	Acme Widget
Mobile	Acme Widget
Book	Johns
Pen	Johns
Pencil	Rollance

SQL Additions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Administration Schemas

No object selected

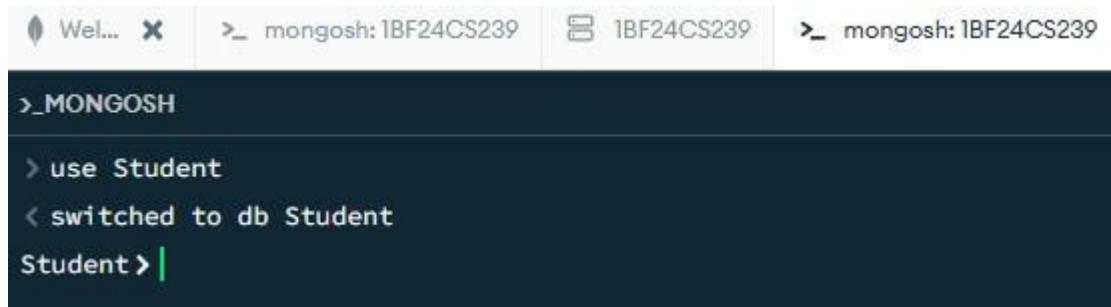
Action Output

Time	Action	Message	Duration / Fetch
19 08:35:41	SELECT p.pname, s.name FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.cid WHERE c.cost < (SELECT MAX(c2.co... 6 rows returned		0.000 sec / 0.000 sec
20 08:37:58	SELECT c.name FROM Supplier c WHERE NOT EXISTS (SELECT p FROM Parts p WHERE p.pid NOT IN (... 2 rows returned		0.000 sec / 0.000 sec
21 08:38:13	SELECT c.name FROM Supplier c WHERE NOT EXISTS (SELECT p FROM Parts p WHERE p.color = 'Red' AND p.pid NOT IN (... 2 rows returned		0.000 sec / 0.000 sec
22 08:38:31	SELECT p.pname FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.cid WHERE s.name = 'Acme Widget' AND p.pid NOT ... 1 rows returned		0.000 sec / 0.000 sec
23 08:38:48	SELECT DISTINCT cl.sid FROM Catalog cl WHERE cl.cost > (SELECT AVG(c2.cost)) FROM Catalog c2 WHERE c2.pid = cl.pid LIMIT 0... 1 rows returned		0.000 sec / 0.000 sec
24 08:39:03	SELECT p.pname, s.name FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.cid WHERE c.cost < (SELECT MAX(c2.co... 6 rows returned		0.000 sec / 0.000 sec

Object Info Session

Experiment no8 : Nosql lab1(Student)

Create a database “Student” with the following attributes Rollno, Age, ContactNo, EmailId.



```
Wel... X >_ mongosh:1BF24CS239 1BF24CS239 >_ mongosh:1BF24CS239
>_MONGOSH
> use Student
< switched to db Student
Student>
```

Insert appropriate values.

```
switched to db Students_Table
Students_Table> db.student.insertMany([
...   { Rollno: 10, Age: 20, ContactNo: "9876543210", EmailId: "rahull0@gmail.com", Name: "RAHUL" },
...   { Rollno: 11, Age: 21, ContactNo: "9123456780", EmailId: "raj11@gmail.com", Name: "RAJ" },
...   { Rollno: 12, Age: 22, ContactNo: "9988776655", EmailId: "rajvardhan12@gmail.com", Name: "RAJVARDHAN" }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6927c9b61771b37f9963b112'),
    '1': ObjectId('6927c9b61771b37f9963b113'),
    '2': ObjectId('6927c9b61771b37f9963b114')
  }
}
```

The screenshot shows the MongoDB Compass interface with the 'student' collection selected. There are three documents listed:

- Document 1:**

```
_id: ObjectId('6927c6f526e0825c3463b112')
Rollno : 10
Age : 20
ContactNo : "9876543210"
EmailId : "newemail10@gmail.com"
Name : "ABC"
```
- Document 2:**

```
_id: ObjectId('6927c6f526e0825c3463b113')
Rollno : 11
Age : 21
ContactNo : "9123456780"
EmailId : "abc11@gmail.com"
Name : "ABC"
```
- Document 3:**

```
_id: ObjectId('6927c6f526e0825c3463b114')
Rollno : 12
Age : 22
ContactNo : "9988776655"
EmailId : "abc12@gmail.com"
Name : "XYZ"
```

At the bottom left, there is a command-line interface window showing a MongoDB query:

```
Students_Table> db.student.updateOne(
...   { Rollno: 10 },
...   { $set: { EmailId: "raman10@gmail.com" } }
... )
```

Write query to update Email-Id of a student with rollno 10.

```
Students_Table> db.student.updateOne(
...   { Rollno: 10 },
...   { $set: { EmailId: "raman10@gmail.com" } }
... )
```

Replace the student name from “ABC” to “FEM” of rollno 11.

```
Students_Table> db.student.updateOne(  
...   { Rollno: 10 },  
...   { $set: { EmailId:"raman10@gmail.com" } }  
... )  
...  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}  
Students_Table> db.student.updateOne(  
...   { Rollno: 11, Name: "RAJ" },  
...   { $set: { Name: "RAJU" } }  
... )  
...  
{  
  acknowledged: true,  
  insertedId: null,
```

Import a given csv dataset from local file system into mongodb collection.

```
mongoexport --db=Student --collection=students --out=students.json
```

Export the created table into local file system.

```
2025-02-20T16:22:13.543+0530      connected to: localhost  
2025-02-20T16:22:13.678+0530      exported 3 records
```

Experiment no9 : Nosql lab1(Customer)

Create a collection by name **Customers** with the following attributes. **Cust_id**, **Acc_Bal**, **Acc_Type**

```
 Welcome  >_ mongosh:1BF24CS239  1BF24CS239  >_ mongosh:1BF24CS239
>_MONGOSH

> use CustomerDB

< switched to db CustomerDB
> db.Customers.insertMany([
  { Cust_id: 101, Acc_Bal: 1500, Acc_Type: "Z" },
  { Cust_id: 102, Acc_Bal: 900, Acc_Type: "A" },
  { Cust_id: 101, Acc_Bal: 1800, Acc_Type: "Z" },
  { Cust_id: 103, Acc_Bal: 1300, Acc_Type: "Z" },
  { Cust_id: 104, Acc_Bal: 700, Acc_Type: "B" }
])

< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693a3695a2d039c4696aa0a8'),
    '1': ObjectId('693a3695a2d039c4696aa0a9'),
    '2': ObjectId('693a3695a2d039c4696aa0aa'),
    '3': ObjectId('693a3695a2d039c4696aa0ab'),
    '4': ObjectId('693a3695a2d039c4696aa0ac')
  }
}
```

Write a query to display those records whose total account balance is greater than 1200 of account type 'Z' for each customer_id.

```
 Welcome  >_ mongosh:1BF24CS239  1BF24CS239  >_ mongosh:1BF24CS239

>_MONGOSH
}
}
> db.Customers.find(
  { Acc_Type: "Z", Acc_Bal: { $gt: 1200 } }
)
< [
  {
    _id: ObjectId('693a3695a2d039c4696aa0a8'),
    Cust_id: 101,
    Acc_Bal: 1500,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('693a3695a2d039c4696aa0aa'),
    Cust_id: 101,
    Acc_Bal: 1800,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('693a3695a2d039c4696aa0ab'),
    Cust_id: 103,
    Acc_Bal: 1300,
    Acc_Type: 'Z'
  }
]
```

Determine Minimum and Maximum account balance for each customer_id.

```
< Welcome > mongosh:1BF24CS239 < 1BF24CS239 > mongosh:1BF24CS239

>_MONGOSH
    Acc_Type: 'Z'
}
{
  _id: ObjectId('693a3695a2d039c4696aa0ab'),
  Cust_id: 103,
  Acc_Bal: 1300,
  Acc_Type: 'Z'
}
> db.Customers.aggregate([
  {
    $group: {
      _id: "$Cust_id",
      Min_Balance: { $min: "$Acc_Bal" },
      Max_Balance: { $max: "$Acc_Bal" }
    }
  }
])
< {
  _id: 104,
  Min_Balance: 700,
  Max_Balance: 700
}
{
  _id: 102,
  Min_Balance: 900,
  Max_Balance: 900
}
{
  _id: 101,
  Min_Balance: 1500,
  Max_Balance: 1800
}
{
  _id: 103,
  Min_Balance: 1300,
  Max_Balance: 1300
}
```

Import a given csv dataset from local file system into mongodb collection

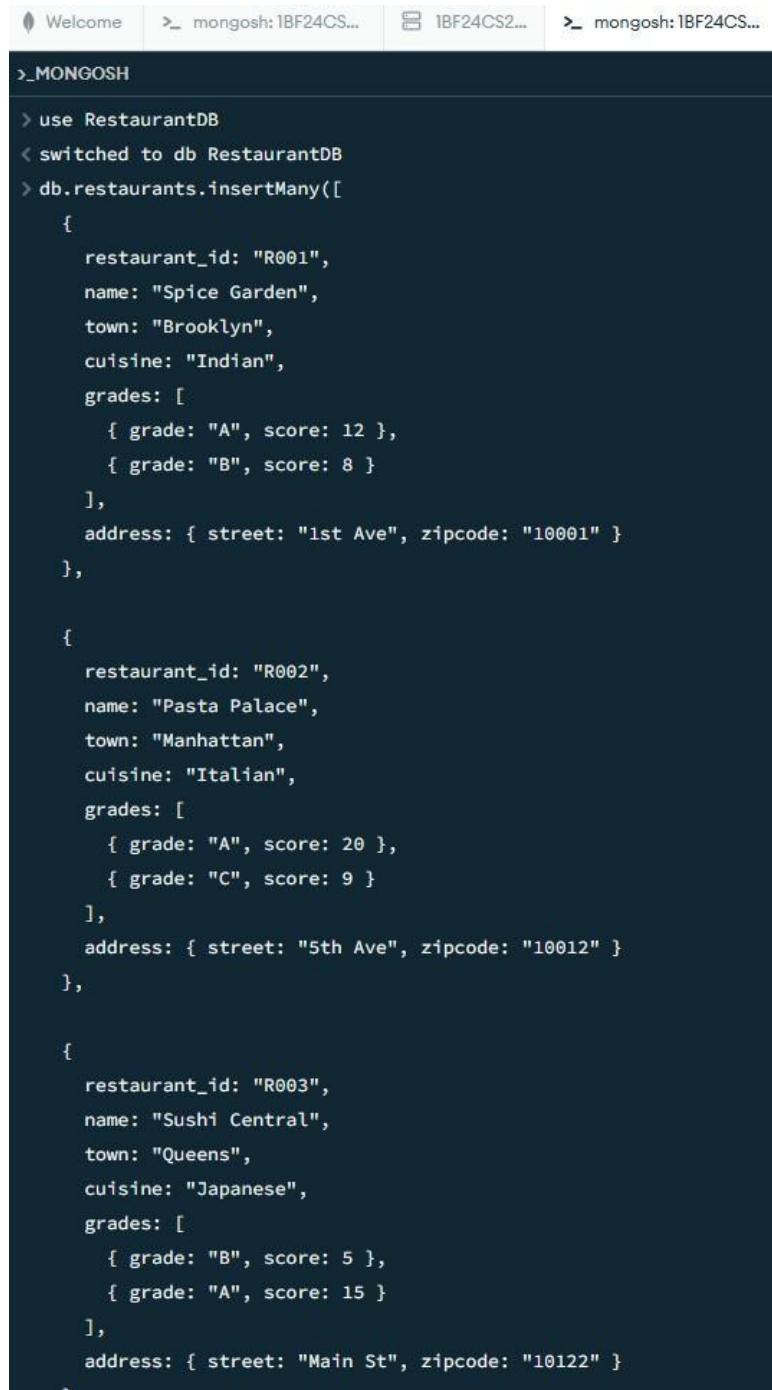
```
mongoexport --db=CustomerDB --collection=Customers --out=customers.json
```

Export the created collection into local file system.

```
2025-02-20T16:55:12.543+0530      connected to: localhost
2025-02-20T16:55:12.684+0530      exported 5 records
```

Experimentno 10: NoSQL Restaurant Database

Write NoSQL Queries on “Restaurant” collection.



The screenshot shows the MongoDB shell interface with the title bar "mongosh:1BF24CS..." and tabs "1BF24CS2..." and "mongosh:1BF24CS...". The command line starts with ">_MONGOSH" and then proceeds to insert three documents into the "RestaurantDB" database's "restaurants" collection. Each document represents a restaurant with fields like "restaurant_id", "name", "town", "cuisine", "grades", and "address".

```
>_MONGOSH
> use RestaurantDB
< switched to db RestaurantDB
> db.restaurants.insertMany([
  {
    restaurant_id: "R001",
    name: "Spice Garden",
    town: "Brooklyn",
    cuisine: "Indian",
    grades: [
      { grade: "A", score: 12 },
      { grade: "B", score: 8 }
    ],
    address: { street: "1st Ave", zipcode: "10001" }
  },
  {
    restaurant_id: "R002",
    name: "Pasta Palace",
    town: "Manhattan",
    cuisine: "Italian",
    grades: [
      { grade: "A", score: 20 },
      { grade: "C", score: 9 }
    ],
    address: { street: "5th Ave", zipcode: "10012" }
  },
  {
    restaurant_id: "R003",
    name: "Sushi Central",
    town: "Queens",
    cuisine: "Japanese",
    grades: [
      { grade: "B", score: 5 },
      { grade: "A", score: 15 }
    ],
    address: { street: "Main St", zipcode: "10122" }
  }
])
```

Write a MongoDB query to display all the documents in the collection restaurants.

```
 Welcome  mongosh:1BF24CS...  1BF24CS2...  mongosh:1BF24CS...
>_MONGOSH
},
{
  restaurant_id: "R004",
  name: "Burger Hub",
  town: "Bronx",
  cuisine: "American",
  grades: [
    { grade: "A", score: 18 },
    { grade: "B", score: 10 }
  ],
  address: { street: "Park Lane", zipcode: "10323" }
},
{
  restaurant_id: "R005",
  name: "Falafel House",
  town: "Staten Island",
  cuisine: "Middle Eastern",
  grades: [
    { grade: "B", score: 7 },
    { grade: "B", score: 11 }
  ],
  address: { street: "Forest Ave", zipcode: "10455" }
}
])
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693a40f14783cb34ee3a2ed2'),
    '1': ObjectId('693a40f14783cb34ee3a2ed3'),
    '2': ObjectId('693a40f14783cb34ee3a2ed4'),
    '3': ObjectId('693a40f14783cb34ee3a2ed5'),
    '4': ObjectId('693a40f14783cb34ee3a2ed6')
  }
}
> db.restaurants.find({})
<
```

Write a MongoDB query to find the restaurant Id, name, town and cuisine for those restaurants which achieved a score which is not more than 10.

```
⚡ Welcome ➤ mongosh:IBF24CS... ➤ 1BF24CS2... ➤ mo

>_MONGOSH

{
  grade: 'B',
  score: 10
}
],
address: {
  street: 'Park Lane',
  zipcode: '10323'
}
}
> db.restaurants.find(
  { "grades.score": { $lte: 10 } },
  { _id: 0, restaurant_id: 1, name: 1, town: 1, cuisine: 1 }
)
< [
  {
    restaurant_id: 'R001',
    name: 'Spice Garden',
    town: 'Brooklyn',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 'R002',
    name: 'Pasta Palace',
    town: 'Manhattan',
    cuisine: 'Italian'
  },
  {
    restaurant_id: 'R003',
    name: 'Sushi Central',
    town: 'Queens',
    cuisine: 'Japanese'
  },
  {
    restaurant_id: 'R004',
    name: 'Burger Hub',
    town: 'Bronx',
    cuisine: 'American'
  }
]
```

Write a MongoDB query to find the average score for each restaurant.

```
 Welcome  >_ mongosh:1BF24CS...  1BF24CS2...  mongo
>_MONGOSH
}
{
  restaurant_id: 'R005',
  name: 'Falafel House',
  town: 'Staten Island',
  cuisine: 'Middle Eastern'
}
> db.restaurants.aggregate([
  { $unwind: "$grades" },
  {
    $group: {
      _id: "$name",
      average_score: { $avg: "$grades.score" }
    }
  },
  { $sort: { _id: 1 } }
])
< [
  {
    _id: 'Burger Hub',
    average_score: 14
  }
  {
    _id: 'Falafel House',
    average_score: 9
  }
  {
    _id: 'Pasta Palace',
    average_score: 14.5
  }
  {
    _id: 'Spice Garden',
    average_score: 10
  }
  {
    _id: 'Sushi Central',
    average_score: 10
  }
]
RestaurantDB >
```