

Real Estate database management system

1. Create Table tenants and inserting values:

The screenshot shows a database management tool interface with two panels. The top panel displays SQL queries, and the bottom panel shows the results of these queries.

SQL Queries:

```
1  
2 • create table tenants  
3   (tenant_id int primary key,tenant_name varchar(30) not null,  
4   tenant_address varchar(50),tenant_contact bigint,tenant_location varchar(50));  
  
2 • describe real_estate.tenants;  
  
2 • insert into real_estate.tenants values(101,'sunil','sector20','9820303030','nerul'),(102,'anil','sector10','9830212121','sanpada'),  
3   (103,'ajay','sector15','9930202010','belapur'),(104,'vijay','sector25','9970404040','nerul'),  
4   (105,'akash','sector16','9870445566','vashi'),(106,'vikas','sector17','9930556655','belapur'),  
5   (107,'suresh','sector3','9870121214','khandeshwar');  
6 • select * from real_estate.tenants;
```

Describe Query Results:

Field	Type	Null	Key	Default	Extra
tenant_id	int	NO	PRI	NULL	
tenant_name	varchar(30)	NO		NULL	
tenant_address	varchar(50)	YES		NULL	
tenant_contact	bigint	YES		NULL	
tenant_location	varchar(50)	YES		NULL	

Insert Query Results:

tenant_id	tenant_name	tenant_address	tenant_contact	tenant_location
101	sunil	sector20	9820303030	nerul
102	anil	sector10	9830212121	sanpada
103	ajay	sector15	9930202010	belapur
104	vijay	sector25	9970404040	nerul
105	akash	sector16	9870445566	vashi
106	vikas	sector17	9930556655	belapur
107	suresh	sector3	9870121214	khandeshwar
NULL	NULL	NULL	NULL	NULL

2.Creating Table Owners:

The screenshot displays a database management interface with three panels. The top panel shows the creation of a table named 'owners' with the following SQL command:

```
1 • create table owners(owner_id int primary key,owner_name varchar(50) not null,owner_address varchar(50),owner_contact bigint,  
2   owner_location varchar(50),tenant_id int,foreign key (tenant_id) references tenants(tenant_id));  
3
```

The middle panel shows the command to describe the table:

```
1 • describe real_estate.owners;
```

The bottom panel shows the result grid for the describe command. The table has the following structure:

	Field	Type	Null	Key	Default	Extra
▶	owner_id	int	NO	PRI	NULL	
	owner_name	varchar(50)	NO		NULL	
	owner_address	varchar(50)	YES		NULL	
	owner_contact	bigint	YES		NULL	
	owner_location	varchar(50)	YES		NULL	
	tenant_id	int	YES	MUL	NULL	

The bottom panel also shows the command to insert data into the 'owners' table:

```
1 • insert into real_estate.owners values(1,'abhishek','sector10','8652433885','nerul',101),  
2   (2,'harshad','sector12','9021313131','mankhurd',102),(3,'akshay','sector13','9870141414','seawoods',103),  
3   (4,'prakash','sector21','9634101010','juinagar',104),(5,'ramesh','sector6','9890808080','kurla',105),  
4   (6,'sagar','sector7','9831505050','nerul',106),(7,'amit','sector19','9970151515','panvel',107);
```

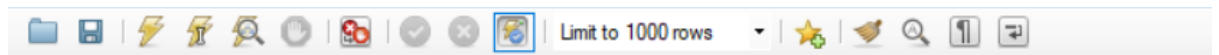
```
1 • SELECT * FROM real_estate.owners;
```

owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id
1	abhishek	sector 10	8652433885	nerul	101
2	harshad	sector 12	9021313131	mankhurd	102
3	akshay	sector 13	9870141414	seawoods	103
4	prakash	sector 21	9634101010	juinagar	104
5	ramesh	sector 6	9890808080	kurla	105
6	sagar	sector 7	9831505050	nerul	106
7	amit	sector 19	9970151515	panvel	107
NULL	NULL	NULL	NULL	NULL	NULL

i. Select, where and in command:

```
6 • select * from real_estate.tenants;  
7 • select tenant_name,tenant_location from real_estate.tenants  
8   where tenant_id IN(101,102,105,106);
```

tenant_name	tenant_location
sunil	nerul
anil	sanpada
akash	vashi
vikas	belapur



```
1 • SELECT owner_name,owner_contact from real_estate.owners where owner_location='nerul';
```

Result Grid		
Filter Rows: <input type="text"/>		
Export:		
Wrap Cell Content:		
	owner_name	owner_contact
▶	abhishek	8652433885
	sagar	9831505050

ii. >,<,<=,>=:




```
1 • SELECT * FROM real_estate.tenants
2 where tenant_name='sunil';
```

Result Grid					
Filter Rows: <input type="text"/>					
Edit:					
Export:					
	tenant_id	tenant_name	tenant_address	tenant_contact	tenant_location
	101	sunil	sector20	9820303030	nerul
	NULL	NULL	NULL	NULL	NULL

```

1 • SELECT * FROM real_estate.owners
2   where owner_location='nerul';

```

Result Grid						
Filter Rows: <input type="text"/>						
Edit: 						
Export/Import:  						
	owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id
▶	1	abhishek	sector10	8652433885	nerul	101
	6	sagar	sector7	9831505050	nerul	106
•	NULL	NULL	NULL	NULL	NULL	NULL





iii. AND, OR & NOT:

AND with in operator:

```

1 • SELECT * FROM real_estate.tenants
2   where tenant_name in('suresh','sunil') and (tenant_location in('khandeshwar','nerul'));

```

Result Grid				
Filter Rows: <input type="text"/>				
Edit: 				
Export/Import:  				
Wrap Cell Content: 				
	tenant_id	tenant_name	tenant_address	tenant_contact
	101	sunil	sector20	9820303030
	107	suresh	sector3	9870121214
	NULL	NULL	NULL	NULL

```

1 • SELECT * FROM real_estate.owners
2   where owner_id in(1,2,3,4) and (tenant_id in(101,102,103,104));

```

Result Grid Filter Rows: <input type="text"/> Edit: Export/Import: Wrap						
	owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id
▶	1	abhishek	sector10	8652433885	nerul	101
	2	harshad	sector12	9021313131	mankhurd	102
	3	akshay	sector13	9870141414	seawoods	103
	4	prakash	sector21	9634101010	juinagar	104
*	NULL	NULL	NULL	NULL	NULL	NULL

OR:

```






1 • SELECT * FROM real_estate.tenants
2   where tenant_location='nerul' or tenant_location='vashi' or tenant_location='belapur';

```

Result Grid Filter Rows: <input type="text"/> Edit: Export/Import: Wrap Cell Content:					
	tenant_id	tenant_name	tenant_address	tenant_contact	tenant_location
▶	101	sunil	sector20	9820303030	nerul
	103	ajay	sector15	9930202010	belapur
	104	vijay	sector25	9970404040	nerul
	105	akash	sector16	9870445566	vashi
	106	vikas	sector17	9930556655	belapur
*	NULL	NULL	NULL	NULL	NULL

NOT:

```
1 • SELECT owner_id,owner_name,owner_contact,owner_location FROM real_estate.owners
2   where not owner_id=6;
```

result Grid			
Filter Rows: <input type="text"/>			
Edit:   			
Export/Import:  			
Wrap Cell Content: <input type="checkbox"/>			
owner_id	owner_name	owner_contact	owner_location
1	abhishek	8652433885	nerul
2	harshad	9021313131	mankhurd
3	akshay	9870141414	seawoods
4	prakash	9634101010	juinagar
5	ramesh	9890808080	kurla
7	amit	9970151515	panvel
NULL	NULL	NULL	NULL

iv. Like Operator:

```
1 • SELECT * FROM real_estate.tenants
2   where tenant_name like '%y';
```

Result Grid					
		Filter Rows:		Edit:	
	tenant_id	tenant_name	tenant_address	tenant_contact	tenant_location
▶	103	ajay	sector15	9930202010	belapur
	104	vijay	sector25	9970404040	nerul
*	NULL	NULL	NULL	NULL	NULL

```
1 • SELECT * FROM real_estate.tenants
2   where tenant_name like '%a%';
```

Result Grid					
		Filter Rows:		Edit:	
	tenant_id	tenant_name	tenant_address	tenant_contact	tenant_location
▶	102	anil	sector10	9830212121	sanpada
	103	ajay	sector15	9930202010	belapur
	104	vijay	sector25	9970404040	nerul
	105	akash	sector16	9870445566	vashi
	106	vikas	sector17	9930556655	belapur
*	NULL	NULL	NULL	NULL	NULL


```

1 • SELECT tenant_id,tenant_name,tenant_address FROM real_estate.tenants
2   where tenant_address like '%sector%';

```

tenant_id	tenant_name	tenant_address
101	sunil	sector20
102	anil	sector10
103	ajay	sector15
104	vijay	sector25
105	akash	sector16
106	vikas	sector17
107	suresh	sector3
NULL	NULL	NULL

v. Between and Not Between:

```

1 • SELECT * from real_estate.owners
2   where owner_location between 'nerul' and 'panvel';

```

owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id
1	abhishek	sector10	8652433885	nerul	101
6	sagar	sector7	9831505050	nerul	106
7	amit	sector19	9970151515	panvel	107
NULL	NULL	NULL	NULL	NULL	NULL

The screenshot shows a database query editor interface. At the top, there is a toolbar with various icons for file operations, execution, and search. Below the toolbar, a SQL query is entered in a text area:

```
1 • SELECT * from real_estate.owners
2   where owner_location not between 'nerul' and 'panvel';
```

Below the query editor, there is a "Result Grid" section. It includes a "Filter Rows:" input field and buttons for "Edit:", "Export/Import:", and "Filter Rows:". The result grid displays the following data:

	owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id
▶	2	harshad	sector12	9021313131	mankhurd	102
	3	akshay	sector13	9870141414	seawoods	103
	4	prakash	sector21	9634101010	juinagar	104
	5	ramesh	sector6	9890808080	kurla	105
*	NULL	NULL	NULL	NULL	NULL	NULL

vi. Distinct & Order by:

```
1 • select distinct owner_id,owner_name,owner_location from real_estate.owners
2 order by owner_name desc;
```


Result Grid			
Filter Rows:			
	owner_id	owner_name	owner_location
▶	6	sagar	nerul
	5	ramesh	kurla
	4	prakash	juinagar
	2	harshad	mankhurd
	7	amit	panvel
	3	akshay	seawoods
	1	abhishek	nerul
*	NULL	NULL	NULL

vii. Null & Not Null:


```
1 • select * from real_estate.owners
2 where owner_location is null;
```

Result Grid						
Filter Rows:						
	owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id
*	NULL	NULL	NULL	NULL	NULL	NULL

viii.Limit & Offset:



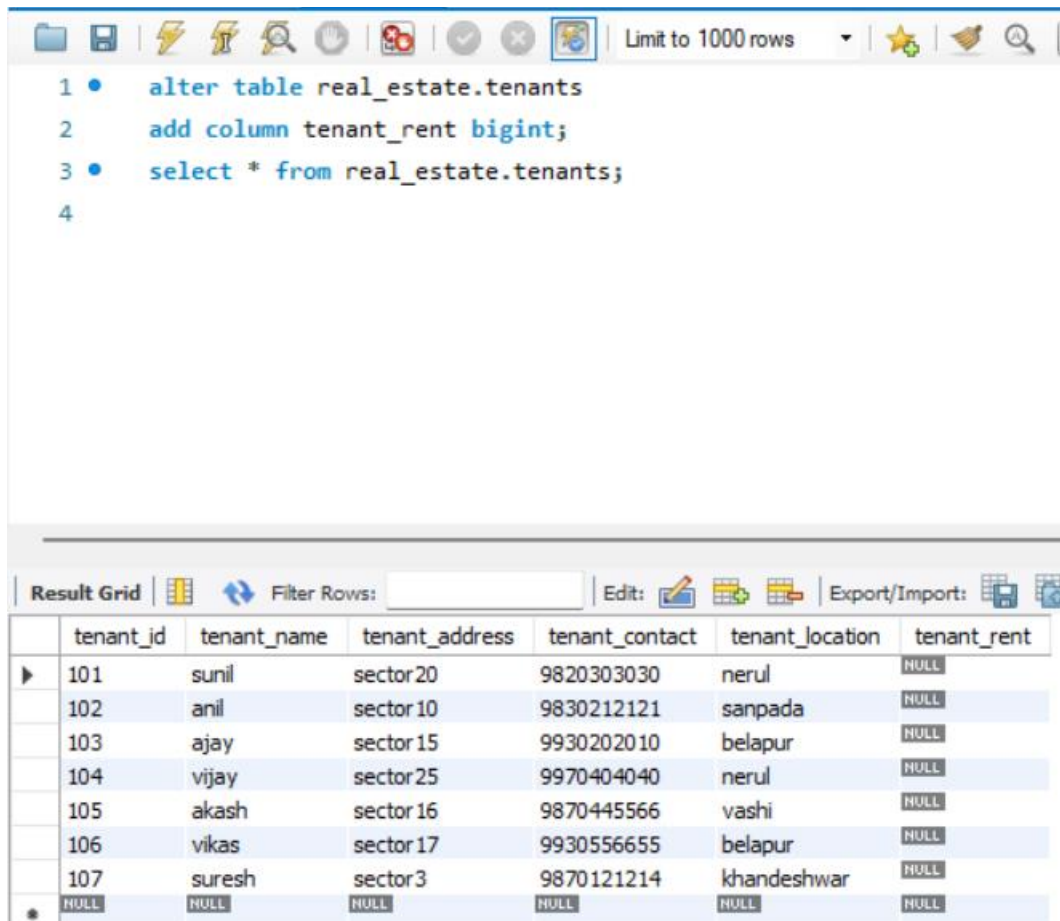
```
1 • select * from real_estate.owners
2 limit 3 offset 3;
```



	owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id
▶	4	prakash	sector21	9634101010	juinagar	104
	5	ramesh	sector6	9890808080	kurla	105
	6	sagar	sector7	9831505050	nerul	106
*	NULL	NULL	NULL	NULL	NULL	NULL

viii. Alter, Update & Case Command:

Alter:



The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search, along with a 'Limit to 1000 rows' dropdown. The SQL editor contains the following commands:

```
1 • alter table real_estate.tenants
2   add column tenant_rent bigint;
3 • select * from real_estate.tenants;
4
```

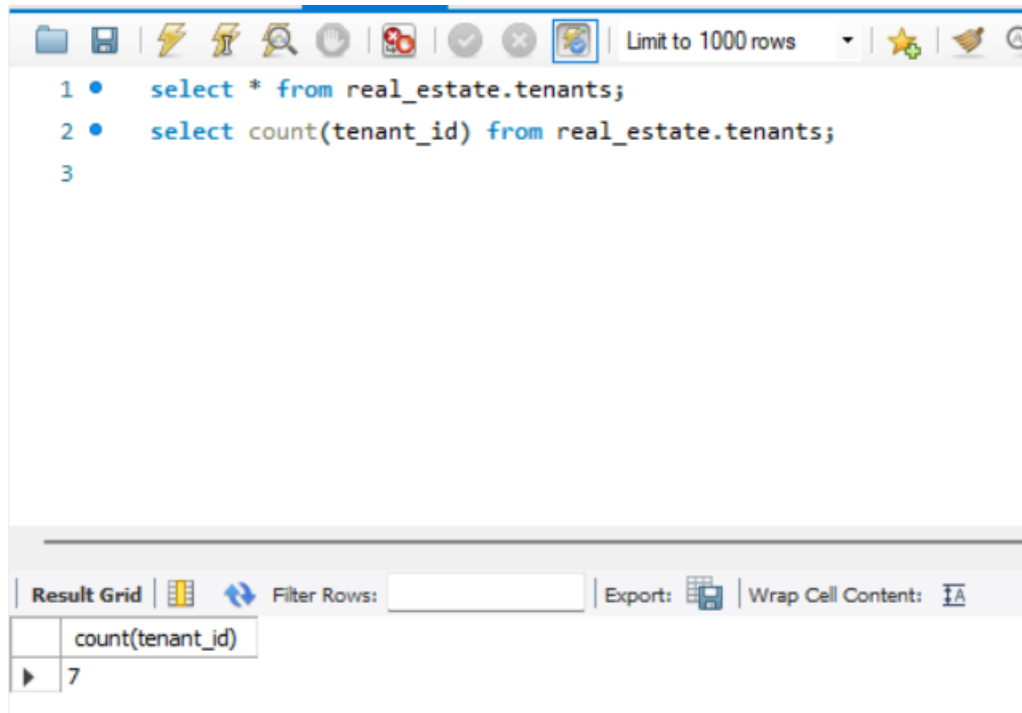
Below the editor is the 'Result Grid' section, which includes a 'Filter Rows' input field and buttons for 'Edit' and 'Export/Import'. The grid displays the following data:

	tenant_id	tenant_name	tenant_address	tenant_contact	tenant_location	tenant_rent
▶	101	sunil	sector20	9820303030	nerul	NULL
	102	anil	sector10	9830212121	sanpada	NULL
	103	ajay	sector15	9930202010	belapur	NULL
	104	vijay	sector25	9970404040	nerul	NULL
	105	akash	sector16	9870445566	vashi	NULL
	106	vikas	sector17	9930556655	belapur	NULL
	107	suresh	sector3	9870121214	khandeshwar	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

Update & Case:

ix. Aggregate Functions (Count, Sum, Min, Max, Avg):

Count():



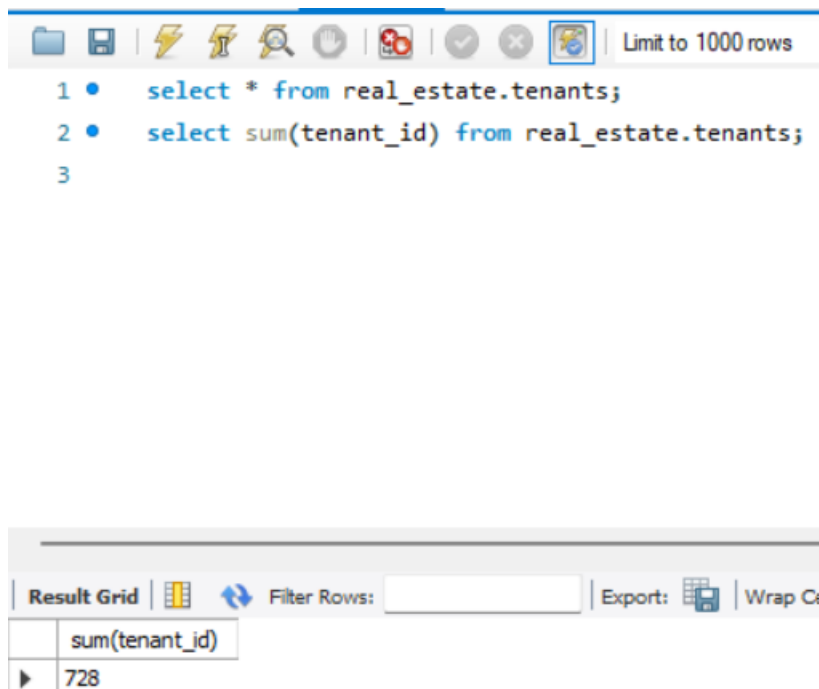
The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following code:

```
1 • select * from real_estate.tenants;  
2 • select count(tenant_id) from real_estate.tenants;  
3
```

Below the editor, the 'Result Grid' tab is active. It displays the result of the second query:

count(tenant_id)
7

Sum():




The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following code:

```
1 • select * from real_estate.tenants;  
2 • select sum(tenant_id) from real_estate.tenants;  
3
```


Below the editor, the 'Result Grid' tab is active. It displays the result of the second query:

sum(tenant_id)
728

Min():




```
1 • select * from real_estate.tenants;
2 • select min(tenant_id) from real_estate.tenants;
3
```




Result Grid	
	min(tenant_id)
▶	101

Max():



```
1 • select * from real_estate.tenants;
2 • select max(tenant_id) from real_estate.tenants;
3
```



Result Grid	
	max(tenant_id)
▶	107

Avg():

```
1 • select * from real_estate.tenants;
2 • select avg(tenant_id) from real_estate.tenants;
3
```

Result Grid | Filter Rows: | Export: | Wrap Cells

avg(tenant_id)
104.0000

x. Delete:

```
1 • select * from real_estate.owners;
2 • Delete from real_estate.owners where owner_id=5;
3 • select * from real_estate.owners;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

	owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id
▶	1	abhishek	sector10	8652433885	nerul	101
	2	harshad	sector12	9021313131	mankhurd	102
	3	akshay	sector13	9870141414	seawoods	103
	4	prakash	sector21	9634101010	juinagar	104
	6	sagar	sector7	9831505050	nerul	106
	7	amit	sector19	9970151515	panvel	107
*	NULL	NULL	NULL	NULL	NULL	NULL

xii. Joins (Inner Join, Left Join, Right Join):

Inner Join:

```
1 • select * from real_estate.owners
2   INNER JOIN real_estate.tenants
3   on real_estate.owners.tenant_id = real_estate.tenants.tenant_id;
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id	tenant_id	tenant_name	tenant_address	tenant_contact	tenant_location	tenant_rent
1	1	abhishek	sector10	8652433885	nerul	101	101	sunil	sector20	9820303030	nerul	15000
2	2	harshad	sector12	9021313131	mankhurd	102	102	anil	sector10	9830212121	sanpada	14000
3	3	akshay	sector13	9870141414	seawoods	103	103	ajay	sector15	9930202010	belapur	16000
4	4	prakash	sector21	9634101010	junagar	104	104	vijay	sector25	9970404040	nerul	10000
6	6	sagar	sector7	9831505050	nerul	106	106	vikas	sector17	9930556655	belapur	14000
7	7	amit	sector19	9970151515	panvel	107	107	suresh	sector3	9870121214	khandeshwar	15000

Left Join:

```
1 • select * from real_estate.owners
2   left JOIN real_estate.tenants
3   on real_estate.owners.tenant_id = real_estate.tenants.tenant_id;
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id	tenant_id	tenant_name	tenant_address	tenant_contact	tenant_location	tenant_rent
1	1	abhishek	sector10	8652433885	nerul	101	101	sunil	sector20	9820303030	nerul	15000
2	2	harshad	sector12	9021313131	mankhurd	102	102	anil	sector10	9830212121	sanpada	14000
3	3	akshay	sector13	9870141414	seawoods	103	103	ajay	sector15	9930202010	belapur	16000
4	4	prakash	sector21	9634101010	juinagar	104	104	vijay	sector25	9970404040	nerul	10000
6	6	sagar	sector7	9831505050	nerul	106	106	vikas	sector17	9930556655	belapur	14000
7	7	amit	sector19	9970151515	panvel	107	107	suresh	sector3	9870121214	khandeshwar	15000

Right Join:

Limit to 1000 rows												
<pre> 1 • select * from real_estate.owners 2 right JOIN real_estate.tenants 3 on real_estate.owners.tenant_id = real_estate.tenants.tenant_id; </pre>												
Result Grid												
Filter Rows: Export: Wrap Cell Content: f												
owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id	tenant_id	tenant_name	tenant_address	tenant_contact	tenant_location	tenant_rent	
1	abhishek	sector10	8652433885	nerul	101	101	sunil	sector20	9820303030	nerul	15000	
2	harshad	sector12	9021313131	mankhurd	102	102	anil	sector10	9830212121	sanpada	14000	
3	akshay	sector13	9870141414	seawoods	103	103	ajay	sector15	9930202010	belapur	16000	
4	prakash	sector21	9634101010	juinagar	104	104	vijay	sector25	9970404040	nerul	10000	
NULL	NULL	NULL	NULL	NULL	NULL	105	akash	sector16	9870445566	vashi	13000	
6	sagar	sector7	9831505050	nerul	106	106	vikas	sector17	9930556655	belapur	14000	
7	amit	sector19	9970151515	panvel	107	107	suresh	sector3	9870121214	khandeshwar	15000	

xiii. Order by and Having:

Order by:

Limit to 1000 rows												
<pre> 1 select * from real_estate.owners 2 order by owner_name asc </pre>												

Result Grid						
Filter Rows: Edit: Export/Import:						
	owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id
▶	1	abhishek	sector10	8652433885	nerul	101
	3	akshay	sector13	9870141414	seawoods	103
	7	amit	sector19	9970151515	panvel	107
	2	harshad	sector12	9021313131	mankhurd	102
	4	prakash	sector21	9634101010	juinagar	104
	6	sagar	sector7	9831505050	nerul	106
*	NULL	NULL	NULL	NULL	NULL	NULL

Limit to 1000 rows

```

1 select * from real_estate.owners
2 order by owner_name desc;

```

Result Grid

	owner_id	owner_name	owner_address	owner_contact	owner_location	tenant_id
▶	6	sagar	sector7	9831505050	nerul	106
	4	prakash	sector21	9634101010	juinagar	104
	2	harshad	sector12	9021313131	mankhurd	102
	7	amit	sector19	9970151515	panvel	107
	3	akshay	sector13	9870141414	seawoods	103
	1	abhishek	sector10	8652433885	nerul	101
*	NULL	NULL	NULL	NULL	NULL	NULL

Having:

Limit to 1000 rows

```

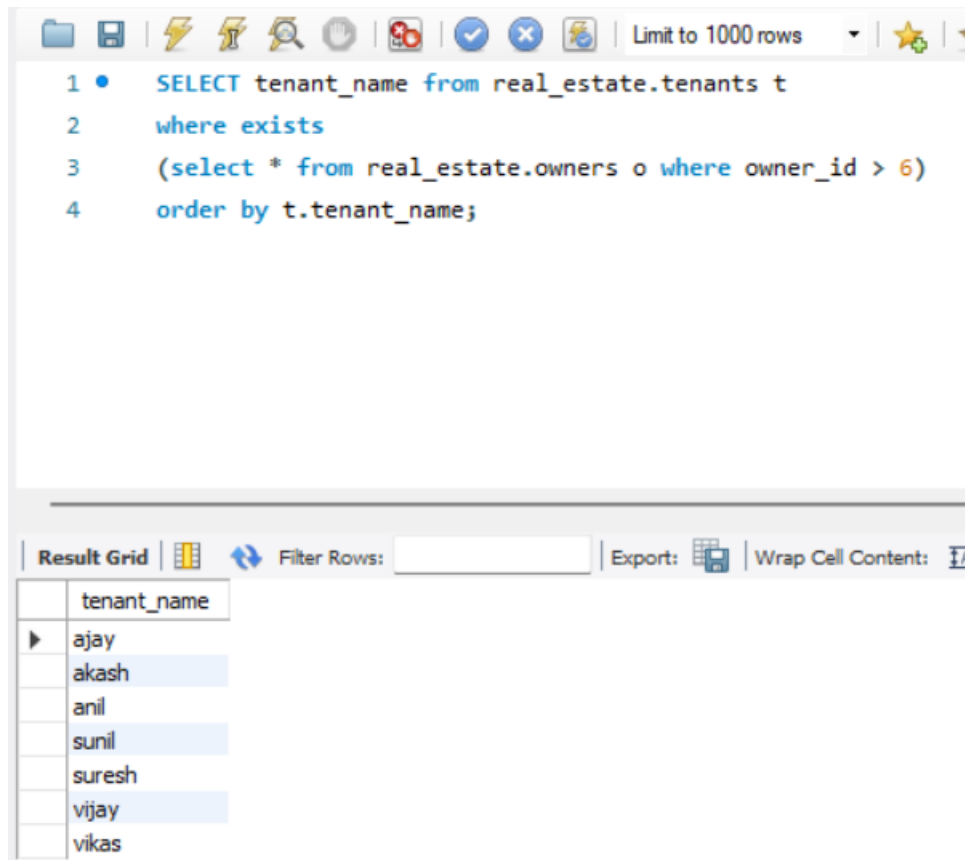
1 • SELECT tenant_id, MIN(tenant_rent)
2 FROM real_estate.tenants
3 GROUP BY tenant_id
4 HAVING MIN(tenant_rent)<15000;

```

Result Grid

	tenant_id	MIN(tenant_rent)
▶	102	14000
	104	10000
	105	13000
	106	14000

xiv. Exists Function:



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, and settings. The query text is as follows:

```
1 • SELECT tenant_name from real_estate.tenants t
2   where exists
3   (select * from real_estate.owners o where owner_id > 6)
4   order by t.tenant_name;
```

Below the query editor, the 'Result Grid' is displayed. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are shown in a table with one column, 'tenant_name', containing eight entries: ajay, akash, anil, sunil, suresh, vijay, and vikas. The rows for 'akash', 'sunil', and 'vijay' are highlighted in blue.

tenant_name
ajay
akash
anil
sunil
suresh
vijay
vikas

xv. Any & All function:

Any:

Limit to 1000 rows

```

1 • select owner_name,owner_contact,owner_location
2   from real_estate.owners
3   where owner_id = any
4     (select owner_id
5      from real_estate.owners
6      where owner_location = 'nerul');

```

Result Grid Filter Rows: Export: Wrap Cell

	owner_name	owner_contact	owner_location
▶	abhishek	8652433885	nerul
	sagar	9831505050	nerul

All:

Limit to 1000 rows ▼

```

1 • select all owner_name,owner_contact,owner_location
2   from real_estate.owners
3   where true;

```

Result Grid Filter Rows: Export: Wrap Cell Co

	owner_name	owner_contact	owner_location
▶	abhishek	8652433885	nerul
	harshad	9021313131	mankhurd
	akshay	9870141414	seawoods
	prakash	9634101010	juinagar
	sagar	9831505050	nerul
	amit	9970151515	panvel

xvi. Date Functions:

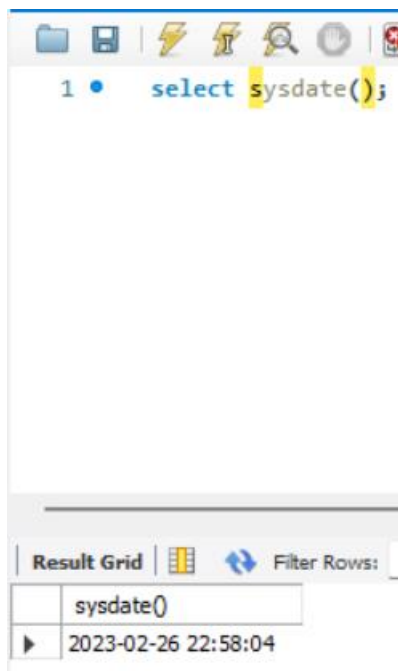
CURDATE():

```
1 • select curdate();
```

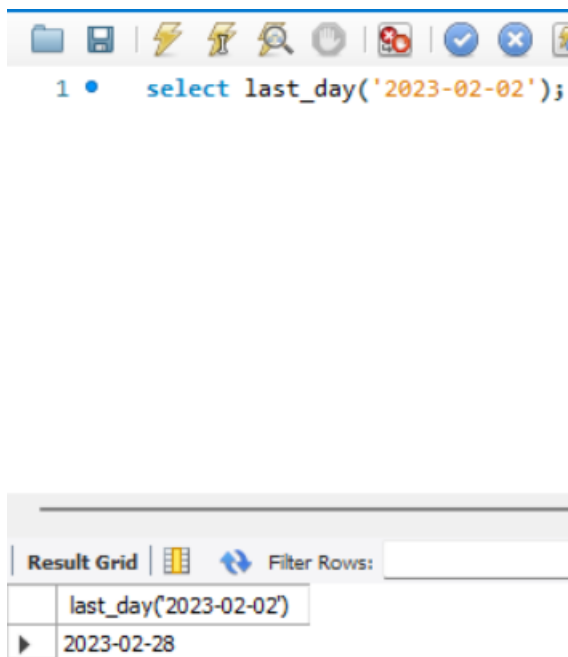
NOW():

1 • `select now();`











SYSDATE():



Last_Day(date):





Date_Format(date,format):

Limit to











1 • `select date_format('2023-02-26', '%Y');`

Result Grid

 Filter Rows:



Export:


	<code>date_format('2023-02-26', '%Y')</code>
▶	2023

Limit to 1000 rows

1 • `select date_format('2023-02-26', '%M %D %Y');`

Result Grid

 Filter Rows:

Export:  Wrap

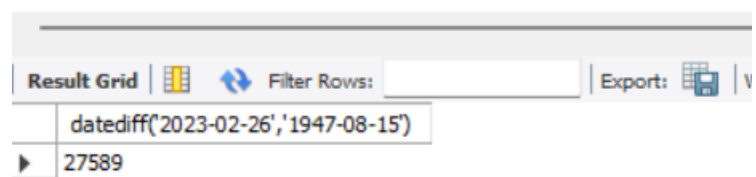
	<code>date_format('2023-02-26', '%M %D %Y')</code>
▶	February 26th 2023

DATEDIFF():



The screenshot shows a toolbar with various icons for file operations, execution, and navigation. Below the toolbar, a single SQL query is entered in a text field.

```
1 • select datediff('2023-02-26','1947-08-15');
```



The screenshot shows a 'Result Grid' with a single row of data. The first column contains the SQL function call, and the second column contains the result of the function.

datediff('2023-02-26','1947-08-15')	27589