

1. Querying Data by Using Joins and Subqueries

Two tables that are going to use:

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
(110, 'Helen Martin', 27, 'Delhi', 64000);

create table test1(project_id int,project_name varchar(50),project_manager varchar(50),id int foreign key references test(emp_id));
insert into test1 values(1,'project A','manoj',101),(2,'project B','sachin',102),(1,'project A','manoj',103),
(3,'project C','hari',104),(5,'project D','sheldon',105),(6,'project E','Nancy',106),(7,'project K','mike',107);

select * from test;
select * from test1;
```

The Results pane displays two tables:

emp_id	emp_name	age	city	salary	
1	101	John	25	Mumbai	60000
2	102	Mike Jane	30	Chennai	75000
3	103	Bob Johnson	25	Salem	70000
4	104	Alice Williams	35	Kanur	80000
5	105	Charlie Brown	22	Delhi	55000
6	106	Eva Davis	29	Kochi	68000
7	107	David Miller	33	Pune	72000
8	108	Grace Wilson	20	Chennai	67000

project_id	project_name	project_manager	id
1	project A	manoj	101
2	project B	sachin	102
3	project A	manoj	103
4	project C	hari	104
5	project D	sheldon	105
6	project E	Nancy	106
7	project K	mike	107

The status bar at the bottom indicates "Query executed successfully."

Inner join: select all rows from both the tables as the condition is satisfied,

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
--Inner join : selects all rows from both the tables as long as the condition is satisfied.
select t.emp_name,t.salary,t.age, t1.project_name from test t inner join test1 t1 on t.emp_id=t1.id;
select t.emp_name,t.salary,t.age, t1.project_name from test t inner join test1 t1 on t.emp_id=t1.id where city='Salem';
```

The Results pane displays the following table:

emp_name	salary	age	project_name
John	60000	25	project A
Mike Jane	75000	30	project B
Bob Johnson	70000	25	project A
Alice Williams	80000	35	project C
Charlie Brown	55000	22	project D
Eva Davis	68000	29	project E
David Miller	72000	33	project K

The status bar at the bottom indicates "Query executed successfully."

Left join: Returns all rows of the left table on the left side of the join and matches rows from right side & non matching rows will show as **NULL**

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
select t.emp_name,t.age, t1.project_name from test t, test1 t1 where t.emp_id=t1.id;
select t.emp_name,t.age, t1.project_name from test t, test1 t1 where t.emp_id<=t1.id;
select t.emp_name,t.age, t1.project_name from test t, test1 t1 where t.emp_id>t1.id;

--Inner join : selects all rows from both the tables as long as the condition is satisfied.
select t.emp_name,t.salary,t.age, t1.project_name from test t inner join test1 t1 on t.emp_id=t1.id;
select t.emp_name,t.salary,t.age, t1.project_name from test t inner join test1 t1 on t.emp_id=t1.id where city='Salem';

--Left join:returns all the rows of the table on the left side of the join and matches rows for the table on the right
select t.emp_name,t.salary,t.age, t1.project_name from test t left join test1 t1 on t.emp_id=t1.id;
```

The Results pane shows the output of the last query (Left Join):

emp_name	salary	age	project_name
John	60000	25	project A
Mike Jane	75000	30	project B
Bob Johnson	70000	25	project A
Alice Williams	80000	35	project C
Charlie Brown	55000	22	project D
Eva Davis	68000	29	project E
David Miller	72000	33	project K
Grace Wilson	67000	20	NULL
Frank Johnson	76000	31	NULL
Helen Martin	64000	27	NULL

The status bar indicates the query was executed successfully, returning 10 rows.

Right join: Returns all rows of the right table on the right side of the join and matches rows from left side & non matching rows will show as **NULL**

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
--Inner join : selects all rows from both the tables as long as the condition is satisfied.
select t.emp_name,t.salary,t.age, t1.project_name from test t inner join test1 t1 on t.emp_id=t1.id;
select t.emp_name,t.salary,t.age, t1.project_name from test t inner join test1 t1 on t.emp_id=t1.id where city='Salem';

--Left join:returns all the rows of the table on the left side of the join and matches rows for the table on the right
select t.emp_name,t.salary,t.age, t1.project_name from test t left join test1 t1 on t.emp_id=t1.id;
select t.emp_name,t1.project_name,t.age, t1.project_name from test t left join test1 t1 on t.emp_id=t1.id
where project_id between 1 and 5;

--Right join:RIGHT JOIN is similar to LEFT JOIN.
select t.emp_name,t.salary,t.age, t1.project_name from test t right join test1 t1 on t.emp_id=t1.id;
```

The Results pane shows the output of the last query (Right Join):

emp_name	salary	age	project_name
John	60000	25	project A
Mike Jane	75000	30	project B
Bob Johnson	70000	25	project A
Alice Williams	80000	35	project C
Charlie Brown	55000	22	project D
Eva Davis	68000	29	project E
David Miller	72000	33	project K

The status bar indicates the query was executed successfully, returning 7 rows.

Full join: combining results from both left and right join

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
--RIGHT JOIN,RIGHT JOIN IS SAME AS LEFT JOIN.
select t.emp_name,t.salary,t.age, t1.project_name from test t right join test1 t1 on t.emp_id=t1.id;
select t.emp_name,t1.project_name,t.age, t1.project_name from test t right join test1 t1 on t.emp_id=t1.id
where city='Chennai';

--FULL JOIN :result-set by combining results of both LEFT JOIN and RIGHT JOIN.
select t.emp_name,t.salary,t.age, t1.project_name from test t full join test1 t1 on t.emp_id=t1.id;
```

The Results pane displays the output of the full join query:

emp_name	salary	age	project_name
John	60000	25	project A
Mike Jane	75000	30	project B
Bob Johnson	70000	25	project A
Alice Williams	80000	35	project C
Charlie Brown	55000	22	project D
Eve Davis	68000	29	project E
David Miller	72000	33	project K
Grace Wilson	67000	20	NULL
Frank Johnson	78000	31	NULL
Helen Martin	64000	27	NULL

The status bar at the bottom indicates "Query executed successfully." and "JONAM\SQLEXPRESS (16.0 RTM) JONAM\jonam (S3) trail 00:00:00 10 rows".

Cross join:

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
--FULL JOIN :result-set by combining results of both LEFT JOIN and RIGHT JOIN.
select t.emp_name,t.salary,t.age, t1.project_name from test t full join test1 t1 on t.emp_id=t1.id;
select t.emp_name,t.city,t.age,t1.project_manager,t1.project_id, t1.project_name from test t full join test1 t1 on t.emp_id=t1.id
where city='Chennai' and salary>10000;

--cross join:
select t.emp_name,t.age,t.salary,t.city,t1.project_name,t1.project_manager, t1.id from test t cross join test1 t1;
```

The Results pane displays the output of the cross join query:

emp_name	age	salary	city	project_name	project_manager	id
John	25	60000	Mumbai	project A	manoj	101
Mike Jane	30	75000	Chennai	project A	manoj	101
Bob Johnson	25	70000	Salem	project A	manoj	101
Alice Williams	35	80000	Karur	project A	manoj	101
Charlie Brown	22	55000	Delhi	project A	manoj	101
Eve Davis	29	68000	kochi	project A	manoj	101
David Miller	33	72000	Pune	project A	manoj	101
Grace Wilson	20	67000	Chennai	project A	manoj	101
Frank Johnson	31	78000	Salem	project A	manoj	101
Helen Martin	27	64000	Delhi	project A	manoj	101
John	25	60000	Mumbai	project B	sachin	102
Mike Jane	30	75000	Chennai	project B	sachin	102
Bob Johnson	25	70000	Salem	project B	sachin	102
Alice Williams	35	80000	Karur	project B	sachin	102
Charlie Brown	22	55000	Delhi	project B	sachin	102
Eve Davis	29	68000	kochi	project B	sachin	102
David Miller	33	72000	Pune	project B	sachin	102
Grace Wilson	20	67000	Chennai	project B	sachin	102

The status bar at the bottom indicates "Query executed successfully." and "JONAM\SQLEXPRESS (16.0 RTM) JONAM\jonam (S3) trail 00:00:00 70 rows".

Subqueries: using in,exists,any

The screenshot shows the SQL Server Enterprise Manager interface. The query editor contains the following SQL code:

```
--subqueries:
select * from test where age in (select age from test where age between 25 and 30);
--returns all where the age between 25-30

select emp_name,emp_id from test where exists (select city from test where city like 'c%');
-- retruns the data the city started as c

select emp_id,salary from test where age = any (select age from test where age= 25);
--returns the rows where any of the age is 25

select sum(salary) as sum_ from test where city in(select city from test where salary >10000);
--returns the total salary where salary > 10000
```

The Results pane shows the following data:

emp_id	emp_name	age	city	salary
101	John	25	Mumbai	60000
102	Mike Jane	30	Chennai	75000
103	Bob Johnson	25	Salem	70000
106	Eva Davis	29	Kochi	68000
110	Helen Martin	27	Delhi	64000

The status bar indicates the query executed successfully, returning 5 rows.

Subqueries: using in,exists,any

The screenshot shows the SQL Server Enterprise Manager interface. The query editor contains the following SQL code:

```
--subqueries:
select * from test where age in (select age from test where age between 25 and 30);
--returns all where the age between 25-30

select emp_name,emp_id from test where exists (select city from test where city like 'c%');
-- retruns the data the city started as c

select emp_id,salary from test where age = any (select age from test where age= 25);
--returns the rows where any of the age is 25

select sum(salary) as sum_ from test where city in(select city from test where salary >10000);
--returns the total salary where salary > 10000
```

The Results pane shows the following data:

emp_id	salary
101	60000
103	70000

The status bar indicates the query executed successfully, returning 2 rows.

Subqueries: using from clause

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
--returns the rows where any of the age is 25
select sum(salary) as sum_ from test where city in(select city from test where salary >10000);
--returns the total salary where salary > 10000

-- by using from clause,
select city,salary from (select city,avg(salary) as salary from test group by city)test
where salary >20000;

-- by using select clause,
select city,(select count(salary) from test where salary >10000) as j from test;

-- by using with,--cte
with high_salary(emp_id,emp_name,salary) as(select emp_id,emp_name,salary from test where salary > 70000)
```

The Results pane shows the output of the query:

city	salary
Chennai	71000
Delhi	59500
Karur	80000
Ischi	80000
Mumbai	60000
Pune	72000
Salem	73000

The status bar indicates the query was executed successfully.

Subqueries: using with clause

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
select city,salary from (select city,avg(salary) as salary from test group by city)test
where salary >20000;

-- by using select clause,
select city,(select count(salary) from test where salary >10000) as j from test;

-- by using with,--cte
with high_salary(emp_id,emp_name,salary) as(select emp_id,emp_name,salary from test where salary > 70000)
select emp_id,emp_name,salary from high_salary;

--2.Manipulate data by using sql commands using groupby and having clause.
```

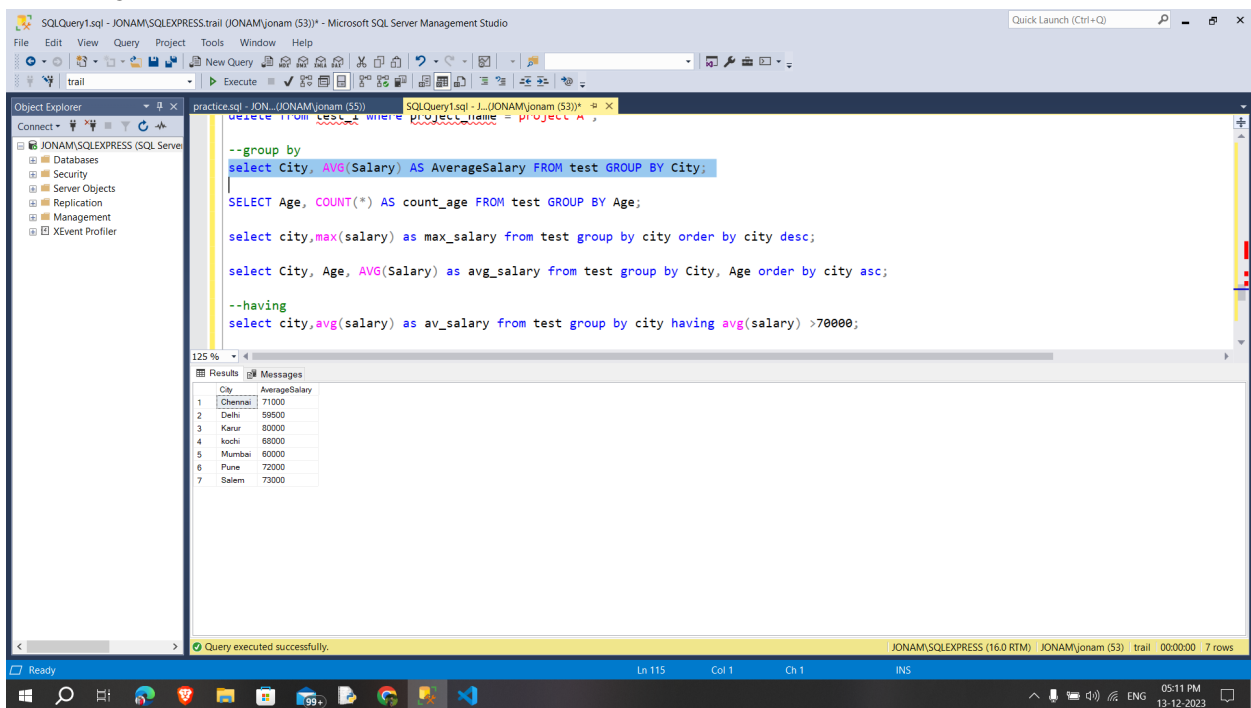
The Results pane shows the output of the query:

emp_id	emp_name	salary
102	Mike Jane	75000
104	Alice Williams	80000
107	David Miller	72000
109	Frank Johnson	76000

The status bar indicates the query was executed successfully.

2.Manipulate data by using sql commands using groupby and having clauses.

Group by:



The screenshot shows the Microsoft SQL Server Management Studio interface. The SQL query editor contains the following code:

```
delete from test1 where project_name = 'project A';

--group by
select City, AVG(Salary) AS AverageSalary FROM test GROUP BY City;

SELECT Age, COUNT(*) AS count_age FROM test GROUP BY Age;

select city,max(salary) as max_salary from test group by city order by city desc;

select City, Age, AVG(Salary) as avg_salary from test group by City, Age order by city asc;

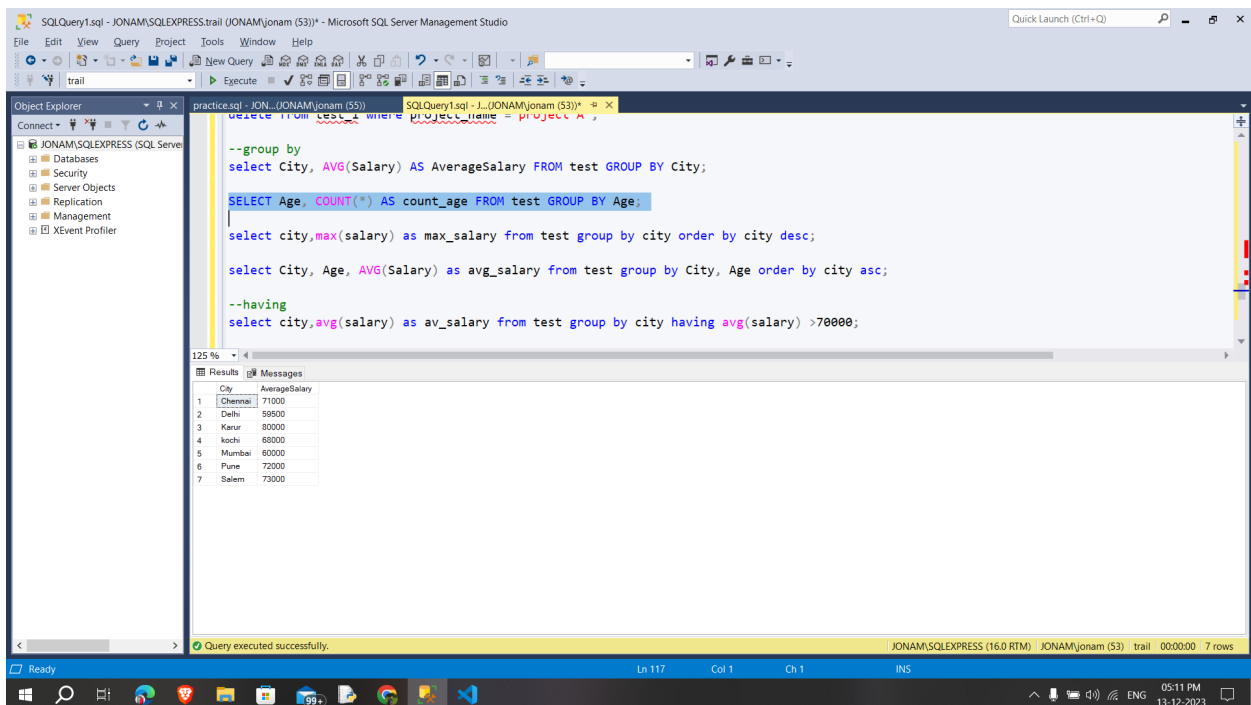
--having
select city,avg(salary) as av_salary from test group by city having avg(salary) >70000;
```

The Results pane displays the output of the first query, showing a table with two columns: City and AverageSalary.

City	AverageSalary
Chennai	71000
Delhi	59500
Kanur	80000
kochi	68000
Mumbai	60000
Pune	72000
Salem	73000

The status bar at the bottom indicates "Query executed successfully." and "JONAM/SQLEXPRESS (16.0 RTM) | JONAM\jonam (53) | trail | 00:00:00 | 7 rows".

Group by:



The screenshot shows the Microsoft SQL Server Management Studio interface. The SQL query editor contains the following code:

```
delete from test1 where project_name = 'project A';

--group by
select City, AVG(Salary) AS AverageSalary FROM test GROUP BY City;

SELECT Age, COUNT(*) AS count_age FROM test GROUP BY Age;

select city,max(salary) as max_salary from test group by city order by city desc;

select City, Age, AVG(Salary) as avg_salary from test group by City, Age order by city asc;

--having
select city,avg(salary) as av_salary from test group by city having avg(salary) >70000;
```

The Results pane displays the output of the first query, showing a table with two columns: City and AverageSalary.

City	AverageSalary
Chennai	71000
Delhi	59500
Kanur	80000
kochi	68000
Mumbai	60000
Pune	72000
Salem	73000

The status bar at the bottom indicates "Query executed successfully." and "JONAM/SQLEXPRESS (16.0 RTM) | JONAM\jonam (53) | trail | 00:00:00 | 7 rows".

Group by:

The screenshot shows the Microsoft SQL Server Management Studio interface. The SQL query editor contains the following code:

```
delete from test1 where project_name = 'Project A';

--group by
select City, AVG(Salary) AS AverageSalary FROM test GROUP BY City;

SELECT Age, COUNT(*) AS count_age FROM test GROUP BY Age;

select city,max(salary) as max_salary from test group by city order by city desc;

select City, Age, AVG(Salary) as avg_salary from test group by City, Age order by city asc;

--having
select city,avg(salary) as av_salary from test group by city having avg(salary) >70000;
```

The Results pane displays the output of the queries:

City	Age	avg_salary
Chennai	20	67000
Chennai	30	75000
Delhi	22	55000
Delhi	27	64000
Karur	35	80000
Kochi	29	66000
Mumbai	25	60000
Pune	33	72000
Salem	25	70000
Salem	31	76000

The status bar at the bottom indicates: Query executed successfully. JONAM\SQLEXPRESS (16.0 RTM) JONAM\jonam (53) trail 00:00:00 10 rows

Having:

The screenshot shows the Microsoft SQL Server Management Studio interface. The SQL query editor contains the following code:

```
select city,max(salary) as max_salary from test group by city order by city desc;

select City, Age, AVG(Salary) as avg_salary from test group by City, Age order by city asc;

--having
select city,avg(salary) as av_salary from test group by city having avg(salary) >70000;

select age,count(*) as age_count from test group by age having count(*) >1;

select city,sum(salary) as total from test group by city having sum(salary) > 140000;

select city,count(*) as city_count from test group by city having count(*)>1;
```

The Results pane displays the output of the queries:

city	av_salary
Chennai	71000
Karur	80000
Pune	72000
Salem	73000

The status bar at the bottom indicates: Query executed successfully. JONAM\SQLEXPRESS (16.0 RTM) JONAM\jonam (53) trail 00:00:00 4 rows

Having :

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for JONAM/SQLEXPRESS. The central query editor contains the following SQL code:

```
select city,max(salary) as max_salary from test group by city order by city desc;

select City, Age, AVG(Salary) as avg_salary from test group by City, Age order by city asc;

--having
select city,avg(salary) as av_salary from test group by city having avg(salary) >70000;

select age,count(*) as age_count from test group by age having count(*) >1;

select city,sum(salary) as total from test group by city having sum(salary) > 140000;

select city,count(*) as city_count from test group by city having count(*)>1;
```

The Results pane at the bottom shows the output of the last query, displaying a table with two columns: age and age_count.

age	age_count
28	2

The status bar at the bottom indicates that the query was executed successfully.

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for JONAM/SQLEXPRESS. The central query editor contains the following SQL code:

```
select city,max(salary) as max_salary from test group by city order by city desc;

select City, Age, AVG(Salary) as avg_salary from test group by City, Age order by city asc;

--having
select city,avg(salary) as av_salary from test group by city having avg(salary) >70000;

select age,count(*) as age_count from test group by age having count(*) >1;

select city,sum(salary) as total from test group by city having sum(salary) > 140000;

select city,count(*) as city_count from test group by city having count(*)>1;
```

The Results pane at the bottom shows the output of the last query, displaying a table with two columns: city and total.

city	total
Chennai	142000
Salem	148000

The status bar at the bottom indicates that the query was executed successfully.