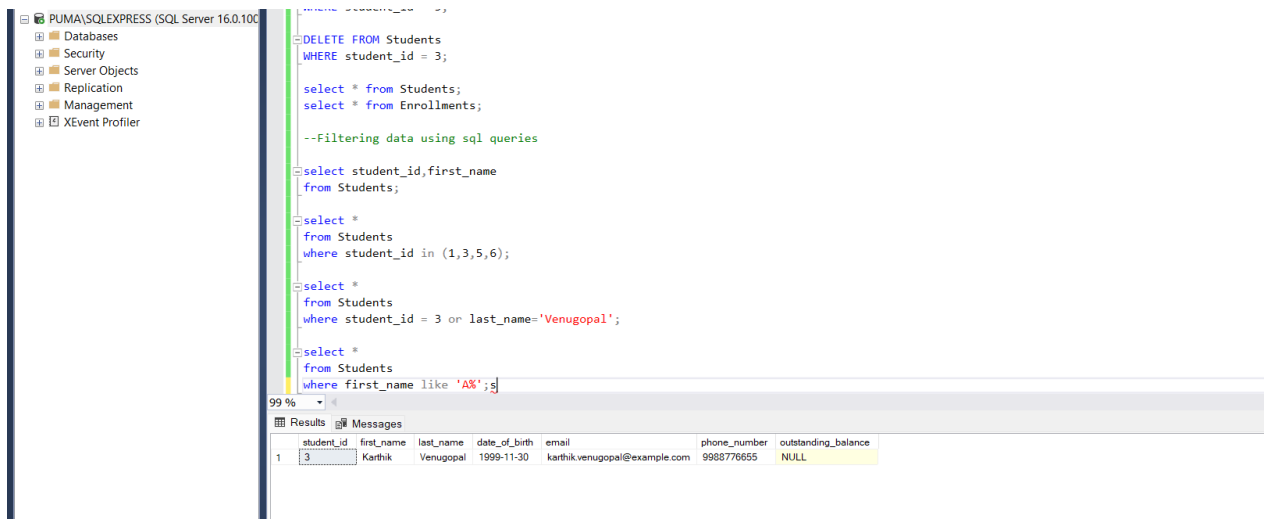


Manipulating Data using DML

Name: Aathirainathan P

Date: 05-11-2024

1. Filtering data using sql queries:



```
--DELETE FROM Students
--WHERE student_id = 3;

select * from Students;
select * from Enrollments;

--Filtering data using sql queries

select student_id,first_name
from Students;

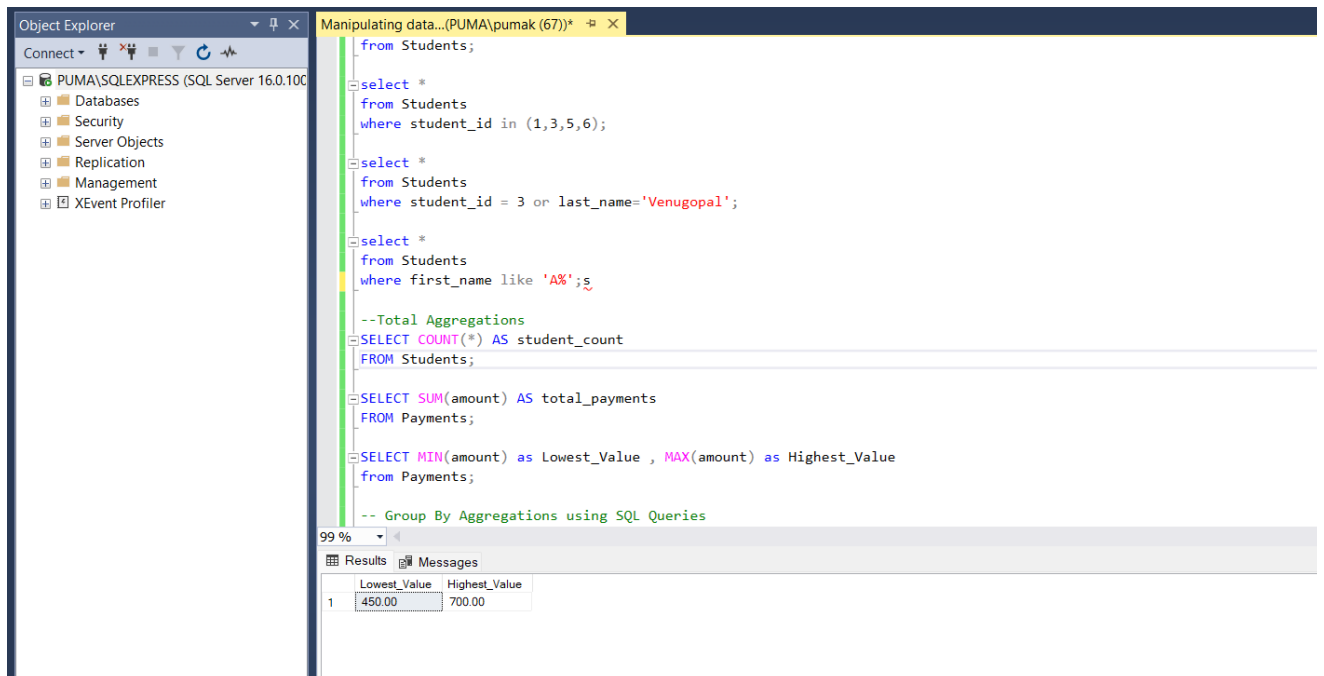
select *
from Students
where student_id in (1,3,5,6);

select *
from Students
where student_id = 3 or last_name='Venugopal';

select *
from Students
where first_name like 'A%';s
```

student_id	first_name	last_name	date_of_birth	email	phone_number	outstanding_balance
3	Karthik	Venugopal	1999-11-30	karthik.venugopal@example.com	9988776655	NULL

2. Total Aggregations:



```
--Total Aggregations
SELECT COUNT(*) AS student_count
FROM Students;

SELECT SUM(amount) AS total_payments
FROM Payments;

SELECT MIN(amount) as Lowest_Value , MAX(amount) as Highest_Value
from Payments;

-- Group By Aggregations using SQL Queries
```

	Lowest_Value	Highest_Value
1	450.00	700.00

3. Group By Aggregations using SQL Queries:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays the server structure for 'PUMA\SQLEXPRESS (SQL Server 16.0.100)'. The main area is a 'Query Window' containing the following SQL code:

```

-- SELECT SUM aggregation using SQL Queries
SELECT SUM(amount) AS total_payments
FROM Payments;

-- SELECT MIN, MAX aggregation using SQL Queries
SELECT MIN(amount) as Lowest_Value , MAX(amount) as Highest_Value
from Payments;

-- Group By Aggregations using SQL Queries
SELECT course_id, COUNT(student_id) AS number_of_students
FROM Enrollments
GROUP BY course_id;

SELECT student_id, SUM(amount) AS total_payment
FROM Payments
GROUP BY student_id;

--OrderBy
select first_name
from Students
order by first_name DESC;

-- Groupby, having, order by
SELECT course_id, COUNT(student_id) AS enrolled_students
FROM Enrollments
GROUP BY course_id
HAVING COUNT(student_id) > 1
  
```

At the bottom, the 'Results' pane shows a table with two columns: 'course_id' and 'number_of_students'. The data is as follows:

course_id	number_of_students
4	2
5	1
6	1
7	1
8	1
9	1
10	1
12	1

4. OrderBy:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the server structure for 'PUMA\SQLEXPRESS (SQL Server 16.0.100)'. The main area is a query window titled 'Manipulating data...(PUMA\pumak (67))' containing the following SQL code:

```
-- Group By Aggregations using SQL Queries
-- SELECT course_id, COUNT(student_id) AS number_of_students
-- FROM Enrollments
-- GROUP BY course_id;

-- SELECT student_id, SUM(amount) AS total_payment
-- FROM Payments
-- GROUP BY student_id;

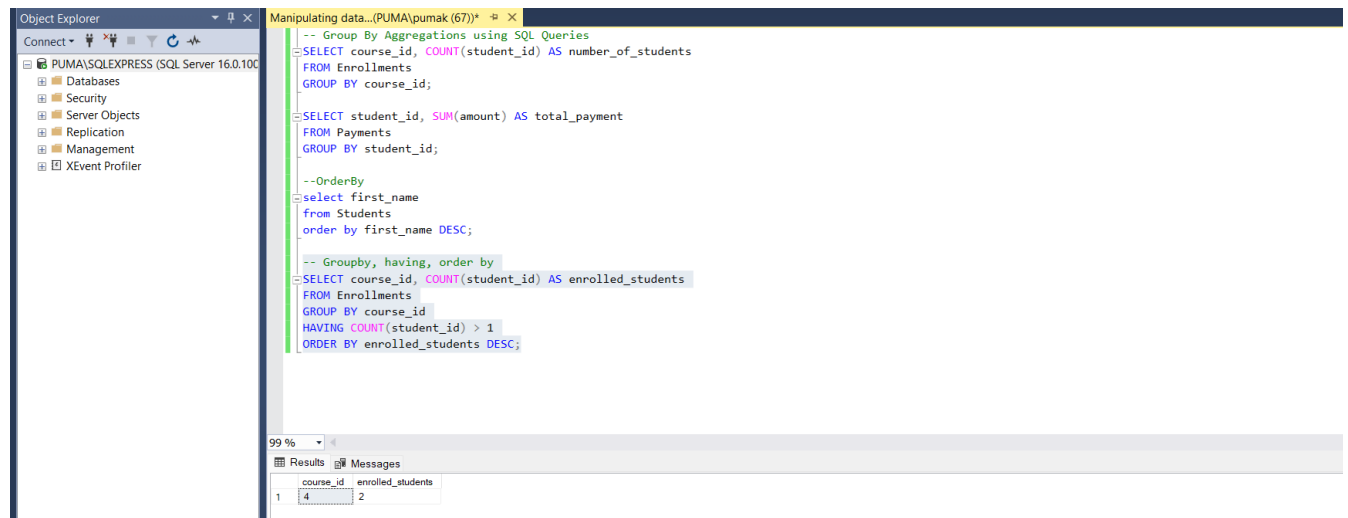
-- OrderBy
select first_name
from Students
order by first_name DESC;

-- Groupby, having, order by
SELECT course_id, COUNT(student_id) AS enrolled_students
FROM Enrollments
GROUP BY course_id
HAVING COUNT(student_id) > 1
ORDER BY enrolled_students DESC;
```

At the bottom, the 'Results' tab is active, displaying a table with 10 rows of student data:

	first_name
1	Vignesh
2	Sowmya
3	Shyam
4	Priya
5	Lakshmi
6	Karthik
7	John
8	Jane
9	Diya
10	Anusha

5. Groupby, having, order by:



The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the server structure for 'PUMA\SQLEXPRESS (SQL Server 16.0.100...)' with folders for Databases, Security, Server Objects, Replication, Management, and XEvent Profiler. The main window is titled 'Manipulating data...(PUMA\pumak (67))' and contains the following SQL code:

```
-- Group By Aggregations using SQL Queries
--SELECT course_id, COUNT(student_id) AS number_of_students
--FROM Enrollments
--GROUP BY course_id;

--SELECT student_id, SUM(amount) AS total_payment
--FROM Payments
--GROUP BY student_id;

--OrderBy
--select first_name
--from Students
--order by first_name DESC;

-- Groupby, having, order by
--SELECT course_id, COUNT(student_id) AS enrolled_students
--FROM Enrollments
--GROUP BY course_id
--HAVING COUNT(student_id) > 1
--ORDER BY enrolled_students DESC;
```

Below the code editor, the 'Results' tab is active, showing a table with two columns: 'course_id' and 'enrolled_students'. The table contains one row of data:

course_id	enrolled_students
4	2