SET Operators in SQL Server

ST1	
id	fname
1	AA
2	BB
3	CC-Duplicate
4	DD
3	CC-Duplicate

ST2	
id	fname
1	AAA
2	BBB
3	CC-Duplicate
6	DD
3	CC-Duplicate

UNION ALL	
id	fname
1	AA
2	BB
3	CC-Duplicate
4	DD
3	CC-Duplicate
1	AAA
2	BBB
3	CC-Duplicate
6	DD
3	CC-Duplicate

UNION	
id	fname
1	AA
1	AAA
2	BB
2	BBB
3	CC-Duplicate
4	DD
6	DD

INTERSECT	
id	fname
3	CC-Duplicate

ST1 EXCEPT ST2	
id	fname
1	AA
2	BB
4	DD

ST2 EXCEPT ST1	
id	fname
1	AAA
2	BBB
6	DD

ST1	
id	fname
1	AA
2	BB
3	CC-Duplicate
4	DD
3	CC-Duplicate

ST2	
id	fname
1	AAA
2	BBB
3	CC-Duplicate
6	DD
3	CC-Duplicate

UNION ALL

id	fname

ST-1

1 — AA
2 — BB
3 - CC-Duplicate
4 - DD
3 - CC-Duplicate

ST-2

1 - AAA
2 - BBB
3 - CC-Duplicate
6 - DD
3 - CC-Duplicate

ST1	
id	fname
1	AA
2	BB
3	CC-Duplicate
4	DD
3	CC-Duplicate

ST2	
id	fname
1	AAA
2	BBB
3	CC-Duplicate
6	DD
3	CC-Duplicate

UNION ALL

id	fname
1	AA
2	BB
3	CC-Duplicate
4	DD
3	CC-Duplicate

ST-1

id	fname
1	AAA
2	BBB
3	CC-Duplicate
6	DD
3	CC-Duplicate

ST-2

UNION

→ 1st find UNION ALL and Duplicate row use only one time.

id	fname
1	AA
2	BB
3	CC-Duplicate
4	DD
1	AAA
2	BBB
6	DD

ST1	
id	fname
1	AA
2	BB
3	CC-Duplicate
4	DD
3	CC-Duplicate

ST2	
id	fname
1	AAA
2	BBB
3	CC-Duplicate
6	DD
3	CC-Duplicate

UNION ALL

id	fname
1	AA
2	BB
3	CC-Duplicate
4	DD
3	CC-Duplicate

ST-1

1	AAA
2	BBB
3	CC-Duplicate
6	DD
3	CC-Duplicate

ST-2

INTERSECT

both table(UNION ALL ans) Duplicate row ans one time.

id	fname
3	CC-Duplicate

ST1	
id	fname
1	AA
2	BB
3	CC-Duplicate
4	DD
3	CC-Duplicate

ST2	
id	fname
1	AAA
2	BBB
3	CC-Duplicate
6	DD
3	CC-Duplicate

UNION ALL

id	fname
1	AA
2	BB
3	CC-Duplicate
4	DD
3	CC-Duplicate

ST-1

id	fname
1	AAA
2	BBB
3	CC-Duplicate
6	DD
3	CC-Duplicate

ST-2

EXCEPT

1st find UNION ALL and remove Duplicate row

Case-1 ST1 EXCEPT ST2

id	fname
1	AA
2	BB
4	DD

Case-2 ST2 EXCEPT ST1

id	fname
1	AAA
2	BBB
6	DD

```sql
CREATE DATABASE SetOperators;

USE SetOperators;
```

```sql
CREATE TABLE ST1(
    id INT,
    fname VARCHAR(15)
);

INSERT INTO ST1 (id,fname)
VALUES
     (1,'AA')
    ,(2,'BB')
    ,(3,'CC-Duplicate')
    ,(4,'DD')
    ,(3,'CC-Duplicate');
```

```sql
CREATE TABLE ST2(
    id INT,
    fname VARCHAR(15)
);

INSERT INTO ST2 (id,fname)
VALUES
     (1,'AAA')
    ,(2,'BBB')
    ,(3,'CC-Duplicate')
    ,(6,'DD')
    ,(3,'CC-Duplicate');
```

```sql
SELECT * FROM ST1;
```

Results | Messages

	id	fname
1	1	AA
2	2	BB
3	3	CC-Duplicate
4	4	DD
5	3	CC-Duplicate

```sql
SELECT * FROM ST2;
```

Results | Messages

	id	fname
1	1	AAA
2	2	BBB
3	3	CC-Duplicate
4	6	DD
5	3	CC-Duplicate

```sql
--UNION ALL
SELECT * FROM ST1
UNION ALL
SELECT * FROM ST2;

--UNION
SELECT * FROM ST1
UNION
SELECT * FROM ST2;

--INTERSECT
SELECT * FROM ST1
INTERSECT
SELECT * FROM ST2;
```

```sql
--EXCEPT
SELECT * FROM ST1
EXCEPT
SELECT * FROM ST2;

--EXCEPT
SELECT * FROM ST2
EXCEPT
SELECT * FROM ST1;
```

SELECT col_1, col_2, col_3 FROM Table-1 ← Query-1

Set operator

SELECT col_1, col_2, col_3 FROM Table-2 ← Query-2

Rule

1. both Query number of column same (total 3 column)

2. both col-1 same data type

Basic Rules for Set Operators

#	Rule	Explanation
1	**Number of Columns must be the same**	Each SELECT in the set operation must return the **same number of columns.**
2	**Data Types must be compatible**	Corresponding columns must have **similar or compatible data types** (e.g., INT with BIGINT, VARCHAR with NVARCHAR).
3	**Column Names are taken from the first query**	The result set's column names come from the **first SELECT statement** only.
4	**Order of Columns must match**	Columns are matched **by position**, not by name. (1st with 1st, 2nd with 2nd, etc.)
5	**Parentheses control precedence**	When multiple set operators are used, **parentheses** determine the order of execution.
6	**ORDER BY only at the end**	ORDER BY can appear **only once**, and it must be **after the final set operation.**
7	**UNION removes duplicates**	Works like UNION ALL followed by DISTINCT.
8	**UNION ALL keeps duplicates**	Faster — includes all rows.
9	**INTERSECT returns common rows**	Shows rows that exist in both results.
10	**EXCEPT returns unique rows from first**	Returns rows in the first query **not** in the second. (MINUS in Oracle.)

```sql
--Rule:1 In-correct
SELECT id,fname FROM ST1
UNION ALL
SELECT id FROM ST2
```

Messages

```
Msg 205, Level 16, State 1, Line 61
All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.
```

```sql
--Rule:1 correct
SELECT id,fname FROM ST1
UNION ALL
SELECT id,fname FROM ST2
```

In simple Rule (1) every query number of total column are Same.

```
--Rule:1 In-correct
SELECT id,fname FROM ST1
UNION ALL
SELECT id FROM ST2
```

→ Query-1 → total 2 column

→ Query-2 → total 1 column

```
--Rule:1 correct
SELECT id,fname FROM ST1
UNION ALL
SELECT id,fname FROM ST2
```

→ Query-1 → total 2 column

→ Query-2 → total 2 column

```sql
--Rule:2 In-correct
SELECT id,fname FROM ST1
UNION ALL
SELECT fname,id FROM ST2
```

28 % No issues found

Results | Messages

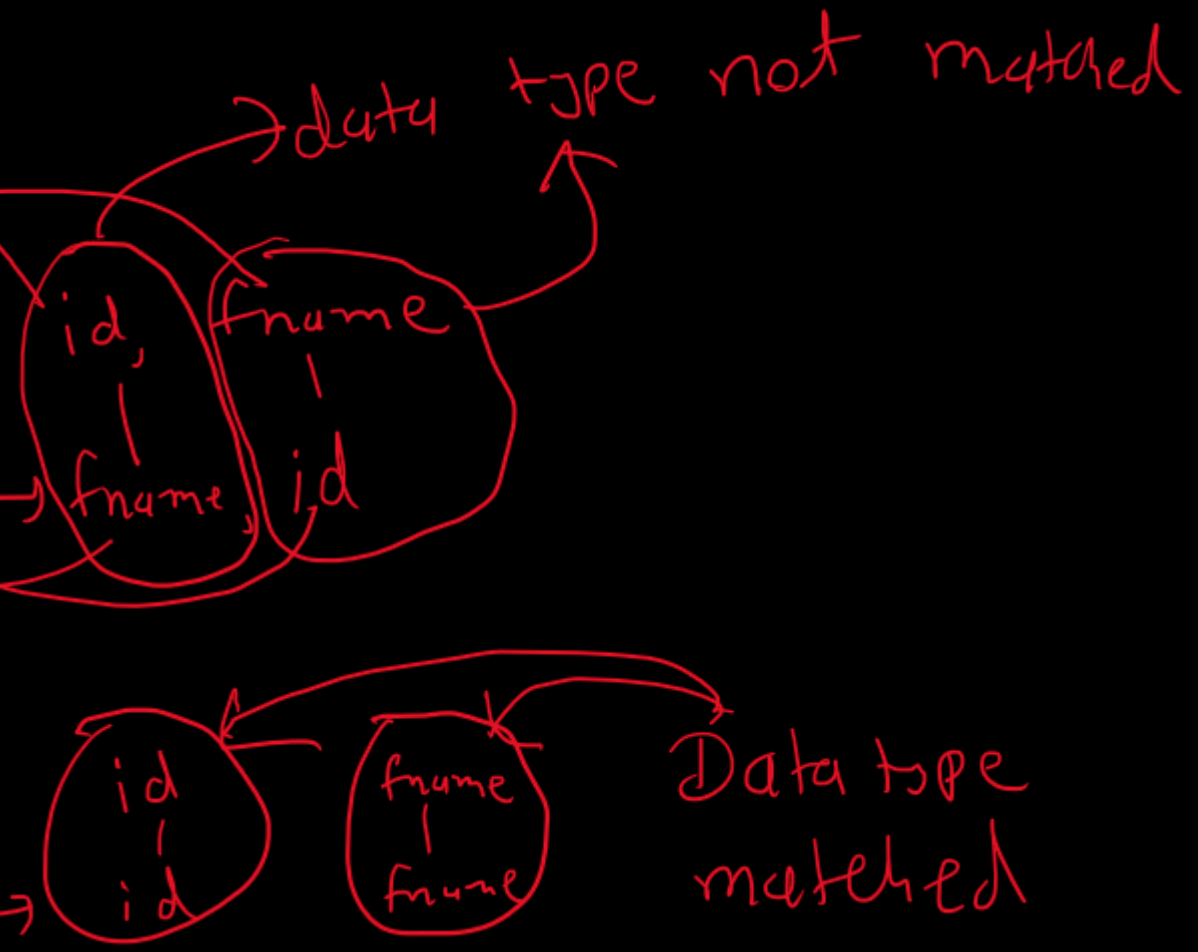
```
Msg 245, Level 16, State 1, Line 72
Conversion failed when converting the varchar value 'AA' to data type int.

Completion time: 2025-10-26T15:48:00.0590390+05:30
```

```sql
--Rule:2 correct
SELECT id,fname FROM ST1
UNION ALL
SELECT id,fname FROM ST2
```

```sql
--Rule:2 In-correct
SELECT id,fname FROM ST1
UNION ALL
SELECT fname,id FROM ST2

--Rule:2 correct
SELECT id,fname FROM ST1
UNION ALL
SELECT id,fname FROM ST2
```



```sql
SELECT * FROM ST1;
```

Results | Messages

	id	fname
1	1	AA
2	2	BB
3	3	CC-Duplicate
4	4	DD
5	3	CC-Duplicate

```sql
SELECT * FROM ST3;
```

Results | Messages

	custId	custName
1	1	AAA
2	2	BBB
3	3	CC-Duplicate
4	6	DD
5	3	CC-Duplicate

```sql
--Rule:3 1st ST1 and 2nd ST3
SELECT * FROM ST1
UNION
SELECT * FROM ST3;
```

```sql
--Rule:3 1st ST3 and 2nd ST1
SELECT * FROM ST3
UNION
SELECT * FROM ST1;
```

Results | Messages

	id	fname
1	1	AA
2	1	AAA
3	2	BB
4	2	BBB
5	3	CC-Duplicate
6	4	DD
7	6	DD

Results | Messages

	custId	custName
1	1	AA
2	1	AAA
3	2	BB
4	2	BBB
5	3	CC-Duplicate
6	4	DD
7	6	DD

```sql
--Rule:6  In-correct
SELECT id,fname FROM ST1
UNION
SELECT custId,custName FROM ST3
ORDER BY custId;
```

```
Msg 207, Level 16, State 1, Line 114
Invalid column name 'custId'.
Msg 104, Level 16, State 1, Line 114
ORDER BY items must appear in the select list if the statement contains a UNION, INTERSECT or EXCEPT operator.
```

```sql
--Rule:6  correct
SELECT id,fname FROM ST1
UNION
SELECT custId,custName FROM ST3
ORDER BY id;
```

As per Rule-3 1st Query Column name in ans

```
--Rule:6  In-correct
SELECT id,fname FROM ST1
UNION
SELECT custId,custName FROM ST3
ORDER BY custId;
```

Here ans id, fname and custId use ORDER BY that is wrong.

```
--Rule:6  correct
SELECT id,fname FROM ST1
UNION
SELECT custId,custName FROM ST3
ORDER BY id;
```

Here ans id, fname and id use ORDER BY that is Correct...

As per above all my Summery

In Short ① every query total number of columns are same.

② every query 1st column same Data types same 2nd col, 3rd col....

③ 1st query column name in final ans.

④ ORDER By use in last and 1st query column use.

⚙️ Set Operator Precedence in SQL Server

When multiple set operators (`UNION`, `UNION ALL`, `INTERSECT`, `EXCEPT`) appear in the same query **without parentheses**,
SQL Server follows a **fixed execution order (precedence)**.

🧩 📘 Precedence Order (Highest → Lowest)

Priority	Operator	Description
1️⃣	**INTERSECT**	Evaluated first — finds common rows between results.
2️⃣	**EXCEPT**	Evaluated after INTERSECT — removes rows found in the second result.
3️⃣	**UNION / UNION ALL**	Evaluated last — combines result sets.

⚠️ Important Notes

- `INTERSECT` has **higher precedence** than both `UNION` and `EXCEPT`.
- `UNION` and `UNION ALL` have **the same precedence** (and are evaluated left to right).
- To **change order**, always use **parentheses**.

🧠 Example 1 (Without Parentheses)

```sql
SELECT id FROM ST1
EXCEPT
SELECT id FROM ST2
INTERSECT
SELECT id FROM ST3;
```

👉 Execution order (by precedence):

```sql
ST2 INTERSECT ST3 → resultA
ST1 EXCEPT resultA
```

So it runs as:

```sql
SELECT id FROM ST1
EXCEPT
(SELECT id FROM ST2 INTERSECT SELECT id FROM ST3);
```

🧠 Example 2 (With Parentheses — Overrides Default)

```sql
(SELECT id FROM ST1
 EXCEPT
 SELECT id FROM ST2)
INTERSECT
SELECT id FROM ST3;
```

👉 Now it runs as:

```sql
(ST1 EXCEPT ST2) → resultA
resultA INTERSECT ST3
```

ST1

id
1
2
3
4
3

ST2

id
1
2
3
6
3

ST3

custId
1
2
3
6
3

```sql
--Example 1 Case: 1
SELECT id FROM ST1
EXCEPT
SELECT id FROM ST2
INTERSECT
SELECT custId FROM ST3;
```

:8 % 2 0

Results Messages

	id
1	4

```sql
--Example 1 Case: 1
SELECT id FROM ST1
EXCEPT
SELECT id FROM ST2
INTERSECT
SELECT custId FROM ST3;
```

1,2,3,4,3

step-1 → 1,2,3,6

step-2 → 4

```sql
--Example 1 Case: 2
SELECT id FROM ST1
EXCEPT
(SELECT id FROM ST2 INTERSECT SELECT custId FROM ST3);
```

Step-1 → 1,2,3,6

1,2,3,4,3

Step-2 → 4

```sql
--Example 2
(SELECT id FROM ST1
 EXCEPT
 SELECT id FROM ST2)
INTERSECT
SELECT custId FROM ST3;
```

:8 % 2 0

Results Messages

id

```sql
--Example 2
(SELECT id FROM ST1
 EXCEPT
 SELECT id FROM ST2)
INTERSECT
SELECT custId FROM ST3;
```

Step-1

→ 1,2,3,4,3

→ 1,2,3,6,3

→ 4

→ 1,2,3,6,3

Step-2 → Not match any value