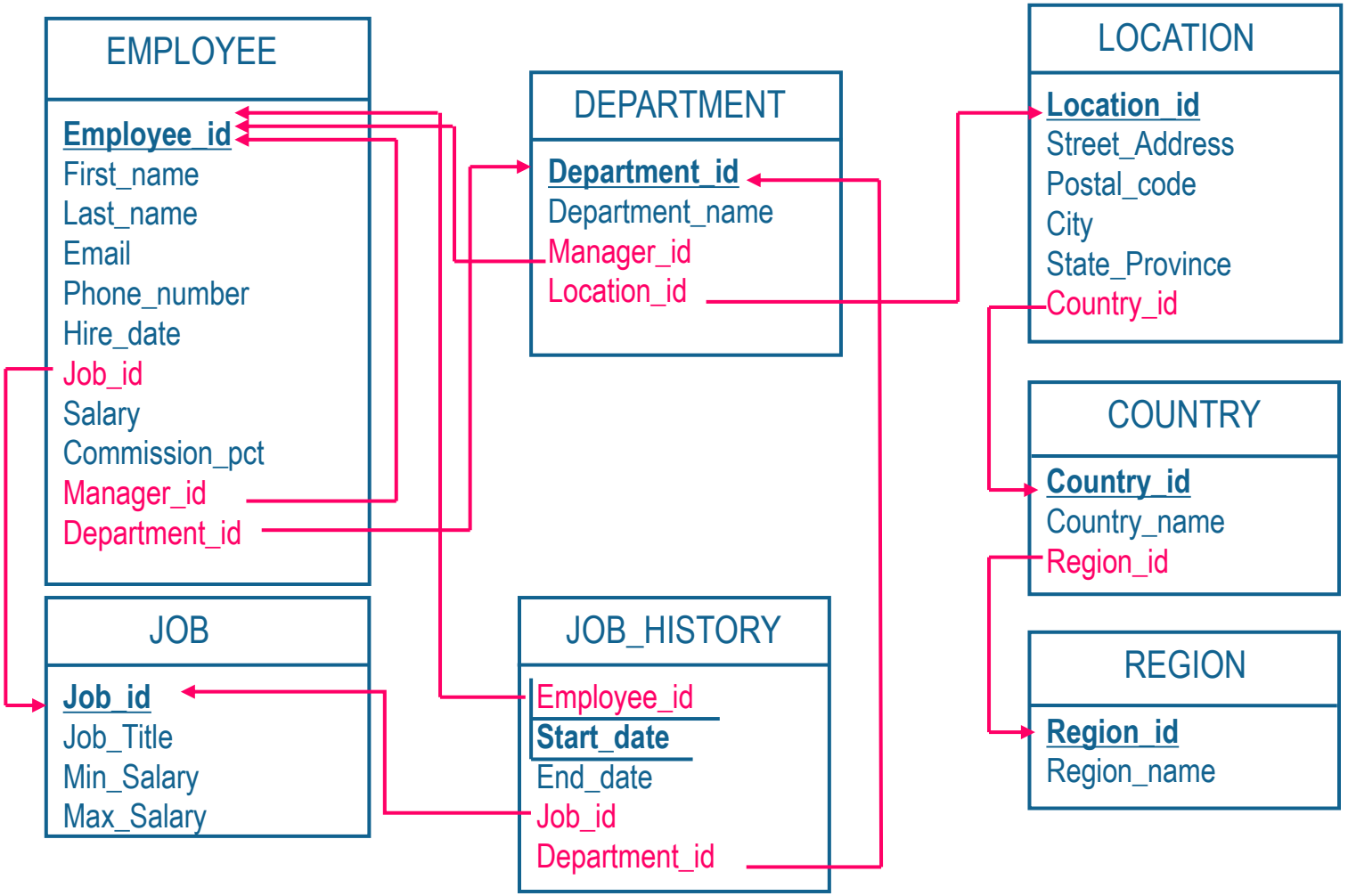
Module 4
Select from Multiple Tables

Table of Content

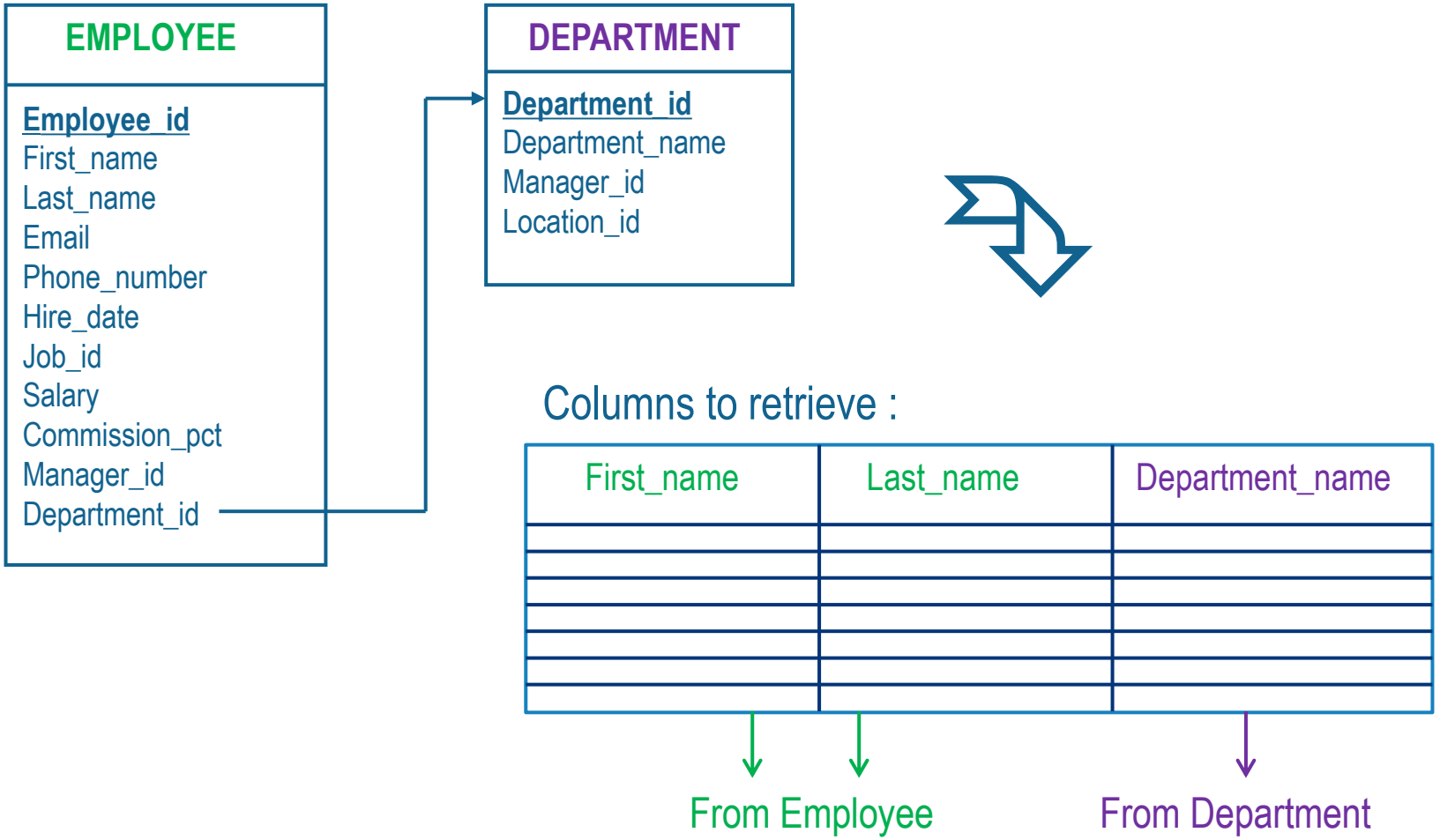
Table of Content

- Non Equijoin
- Outer Join
- Cross Join

Tables Used in Examples



Selecting Data from Multiple Tables



Cartesian Product of Tables

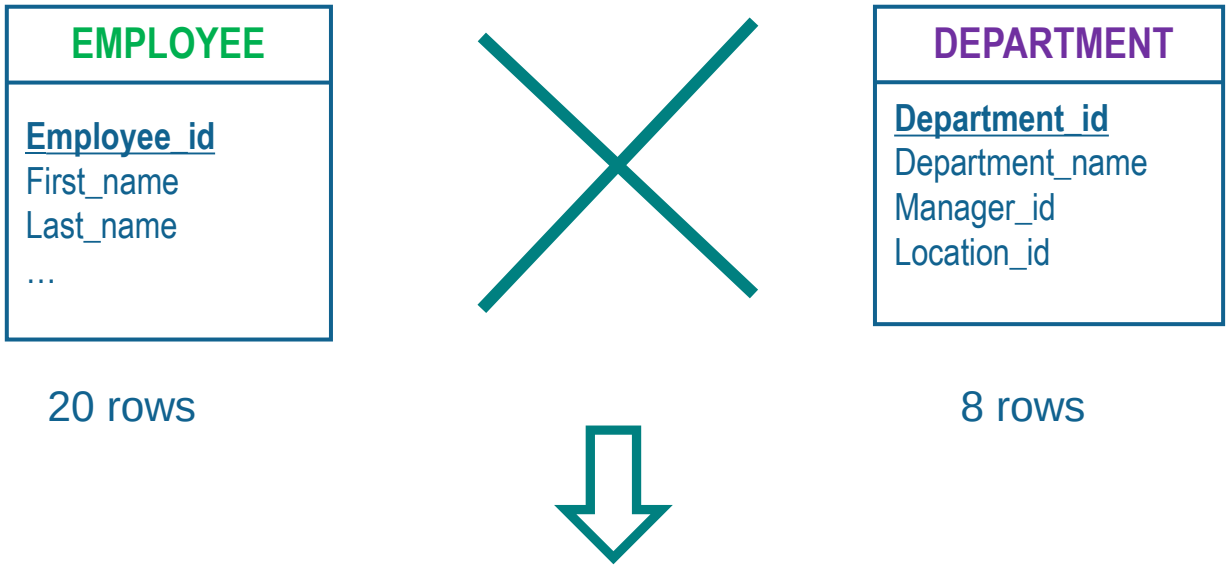
- When a join is made on two tables without any join condition
  - The product cartesian is made
  - i.e, all rows in the first table are joined to all rows in the second table

- A Cartesian product is formed when
  - A join condition is omitted
  - A join condition is invalid
  - E.g.

```
select   last_name, department_name
from     employee, department ;
```

↳No join condition ⇨ cartesian product

Cartesian Product of Tables



Cartesian product

160 rows !!! ⇨not OK

Cartesian Product of Tables

```
select  last_name, employee.department_id, department.department_id, department_name
from    employee, department ;
```

last_name	employee.department_id	department.department_id	department_name
King	90	10	Administration
King	90	20	Marketing
King	90	50	Shipping
King	90	60	IT
King	90	80	Sales
King	90	90	Executive
King	90	110	Accounting
King	90	190	Contracting
Taylor	80	10	Administration
Taylor	80	20	Marketing
Taylor	80	50	Shipping
Taylor	80	60	IT
Taylor	80	80	Sales
Taylor	80	90	Executive
Taylor	80	110	Accounting
Taylor	80	190	Contracting
Mourgos	50	10	Administration
Mourgos	50	20	Marketing
Mourgos	50	50	Shipping
Mourgos	50	60	IT
Mourgos	50	80	Sales
Mourgos	50	90	Executive
Mourgos	50	110	Accounting
Mourgos	50	190	Contracting
...	...	...	...

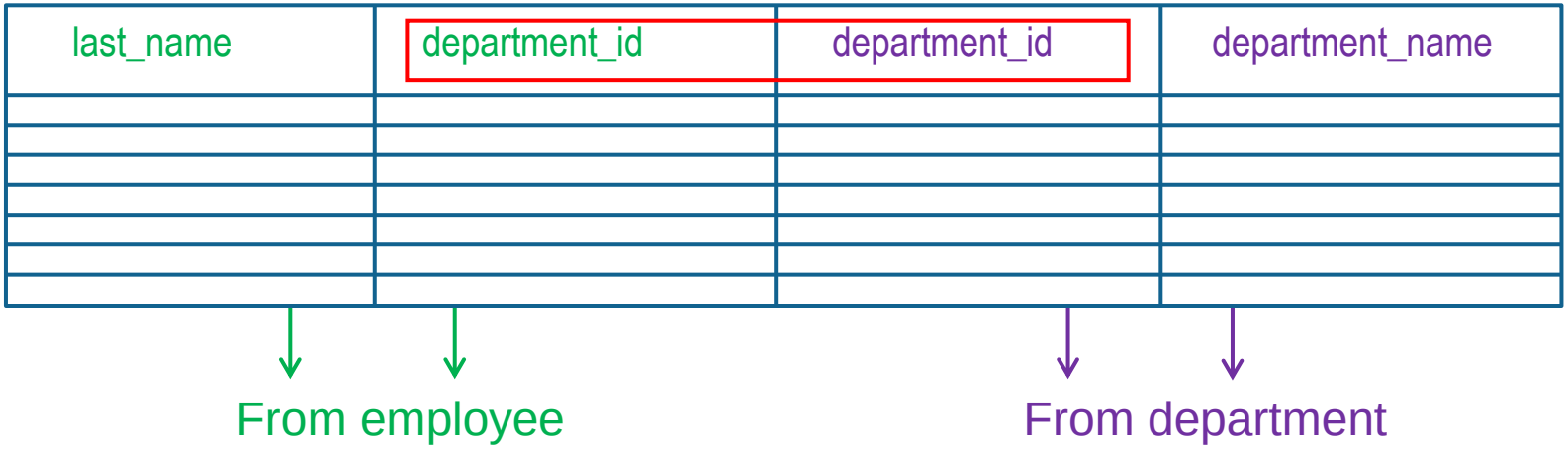
Join Conditions

- To avoid a Cartesian product

  ↳ **Always include a valid join condition !**

- A valid join condition may involve an equality between
  - The foreign key of one table
  - The primary key of the related table

Join Conditions



department_id of employee = department_id of department

Join Conditions

```
SELECT   { table1.column | table2.column, … }
FROM     table1, table2
WHERE    table1.column1 = table2.column2 ;
```

- E.g,

```
select      last_name,department_name
from        employee, department
where       employee.department_id   =  department.department_id ;
```

Foreign key

Primary key

Join condition

Qualifying Ambiguous Column Names

- Add table prefixes to column names that are ambiguous
  - i.e. columns that have the same name in multiple tables
- Use of table prefixes improves performance
- Table aliases can be used as prefixes for column
  - E.g,

```
select    emp.last_name, dep.department_name
from      employee  emp, department  dep
where     emp.department_id   =  dep.department_id ;
```

Oracle SQL >< SQL 99

- SQL 99

```
SELECT      { table1.column | table2.column, … }
FROM        table1
[ NATURAL JOIN table2 ] |
[ JOIN      table2   USING (column_name) ] |
[ JOIN      table2
              ON ( table1.column_name = table2.column_name ) ] |
[ LEFT | RIGHT | FULL OUTER JOIN table2
              ON ( table1.column_name = table2.column_name ) ] |
[ CROSS JOIN table2 ] ;
```

INNER JOIN with the ON Clause

- Use the **ON** clause to specify the two columns to take into account to do the join
  - Usually a foreign key in one table and the primary key of the related table

```
SELECT      { table1.column | table2.column, … }
FROM        table1
INNER JOIN      table2  ON ( table1.column_name  =  table2.column_name ) ;
```

INNER JOIN with the ON Clause

- E.g,

```
select  e.last_name, d.department_name
from    employee e  inner join  department d
 on     (e.department_id = d.department_id);
```

↳ *last name of each employee + name of his department*

```
select  d.department_name, e.last_name
from    employee e inner join  department d
on      (e.employee_id = d.manager_id);
```

↳ *name of each department + last name of its manager*

Join Between N tables

- To join N tables in **SQL Server**
  ↳ use N-1 * (INNER JOIN + ON clauses)
  - For example, a join between 3 tables ⇨ 2 * (INNER JOIN + ON clauses)

- E.g,

```
select    e.last_name, d.department_name, l.city
from          employee   e
inner join    department  d
on        d.department_id = e.department_id
innter join   location  l
on        d.location_id = l.location_id ;
```

Additional Conditions to a Join

- In **SQL Server**
  - Either by adding conditions in the **ON** clause (**AND**)
  - Or by adding a **WHERE** clause

- E.g,

```
select   e.last_name,  d.department_name
from     employee  e   inner join  department  d
on       (e.department_id = d.department_id)
and      e.salary > 5000 ;
```

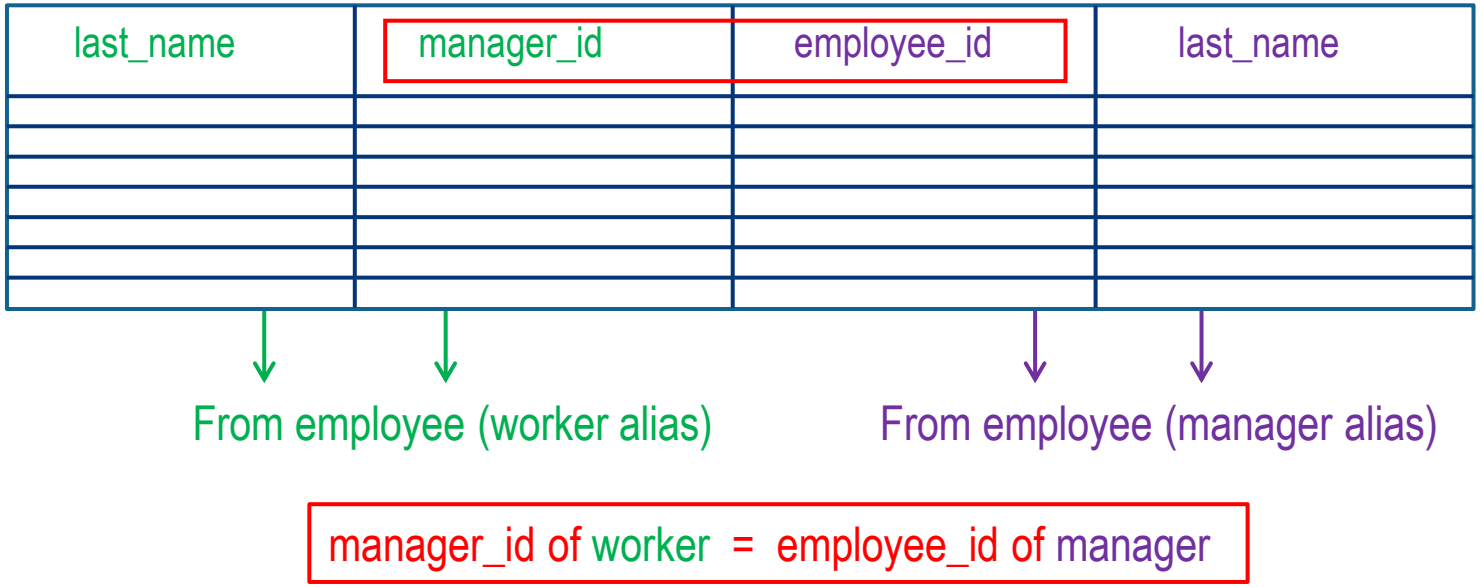
```
select   e.last_name,  d.department_name
from     employee  e   inner join   department  d
on       (e.department_id = d.department_id)
where    e.salary > 5000 ;
```

Joining a Table to Itself – Self Join

E.g,

Select the last name of each employee and the last name of his manager

Joining a Table to Itself – Self Join



Joining a Table to Itself – Self Join

- In Oracle SQL

```
select   worker.last_name, manager.last_name
from     employee worker, employee manager
where    worker.manager_id = manager.employee_id;
```

- In SQL Server

```
select   worker.last_name, manager.last_name
from     employee worker inner join employee manager
on       (worker.manager_id = manager.employee_id);
```

Non Equijoin

- E.g,

  The JOB_GRADE table defines the LOWEST_SAL and HIGHEST_SAL range of values for each GRADE_LEVEL

JOB_GRADE
Grade_level
Lowest_sal
Highest_sal

⇨ *the grade of each employee could be retrieved from his salary*

```
select   e.last_name, e.salary, j.grade_level
from     employee e  inner join  job_grade  j
on       e.salary  between j.lowest_sal  and j.highest_sal ;
```

Outer Join

- Inner Joins return only matched rows
- Outer joins retrieve rows that have no direct match with rows in the other table
  - E.g, *select last name of each employee + name of his department, but including employees with no department*

Outer Join

- Outer joins return the results of the inner join +
  - The unmatched rows from the left table : **LEFT OUTER JOIN**
  - The unmatched rows from the right table : **RIGHT OUTER JOIN**
  - The unmatched rows from the left and right tables : **FULL OUTER JOIN**

```
SELECT    { table1.column | table2.column, … }
FROM      table1
LEFT | RIGHT | FULL  OUTER  JOIN  table2
            ON ( table1.column_name  =  table2.column_name ) ;
```

Outer Join

- Left Outer Join
  - E.g, *select last name and department name of each employee (matched rows) + employees with no department (appearing once)*

```
select   e.last_name, d.department_name
from     employee e  left outer join  department d
         on  (e.department_id = d.department_id) ;
```

- Right Outer Join
  - E.g, *select last name and department name of each employee (matched rows) + departments with no employee (appearing once)*

```
select   e.last_name, d.department_name
from     employee e  right outer join  department d
         on  (e.department_id = d.department_id) ;
```

Outer Join

- Full Outer Join
  - E.g, *select last name and department name of each employee (matched rows)*
    + Employees with no department
    + Departments with no employee

```
select   e.last_name, d.department_name
from     employee  e   full outer join  department d
         on (e.department_id = d.department_id) ;
```

Cross Join

- Cross-product of two tables
  - ↳Cartesian product between the two tables

```
SELECT       { table1.column | table2.column, … }
FROM         table1
CROSS JOIN   table2 ;
```

- E.g.

```
select       last_name, department_name
from         employee
cross join   department ;
```

Summary

Oracle SQL (also SQL Server)

```
SELECT      { column, … }
FROM        { table, … }
WHERE       [ condition(s) ] ;
```

Either join conditions
or normal conditions

Summary

SQL Server

```
SELECT    { table1.column |  table2.column, … }
FROM      table1
[ NATURAL  JOIN  table2] |
[INNER JOIN      table2
            ON ( table1.column_name  =  table2.column_name ) ] |
[ LEFT | RIGHT | FULL  OUTER  JOIN  table2
            ON ( table1.column_name  =  table2.column_name ) ] |
[ CROSS JOIN table2 ]
[ WHERE condition(s) ] ;
```