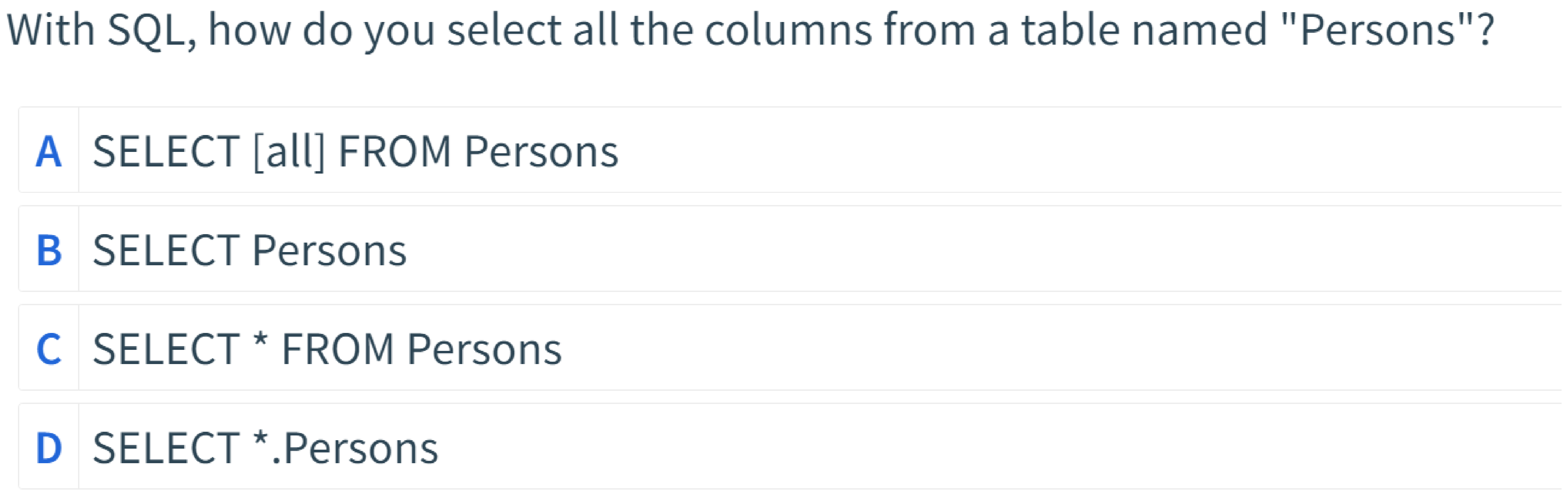
SQL – SQL Joins

Dr. Villanes

Questions
Q1 – Q6

Question 1



With SQL, how do you select all the columns from a table named "Persons"?

A SELECT [all] FROM Persons

B SELECT Persons

C SELECT * FROM Persons

D SELECT *.Persons

Question 2

In the following query, what is the number 2 referring to?

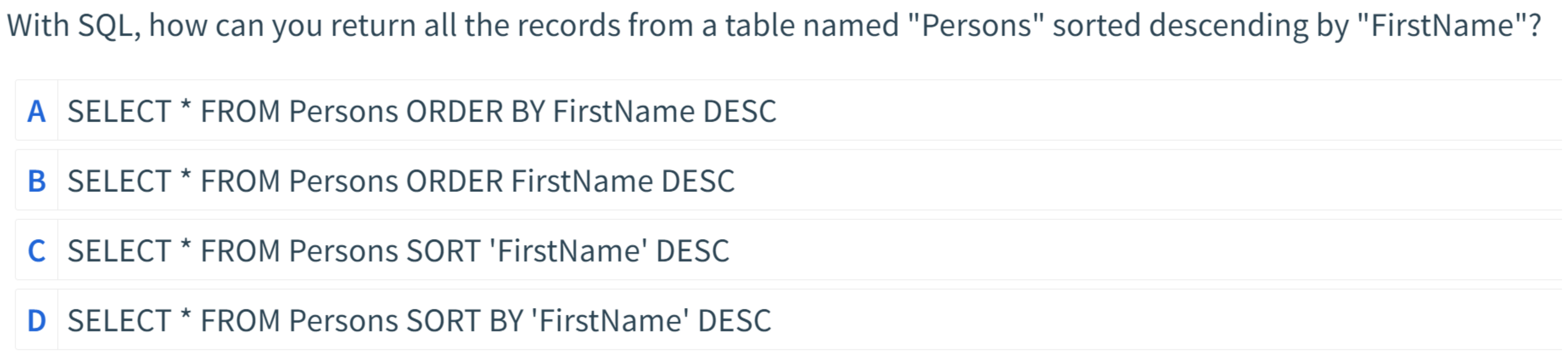
```
select ID, Capital_Gain, capital_gain*0.10
from exercise.records
where capital_gain <> 0
order by 2;
```

A ID

B Capital_Gain

C Capital_Gain*0.10

Question 3



With SQL, how can you return all the records from a table named "Persons" sorted descending by "FirstName"?

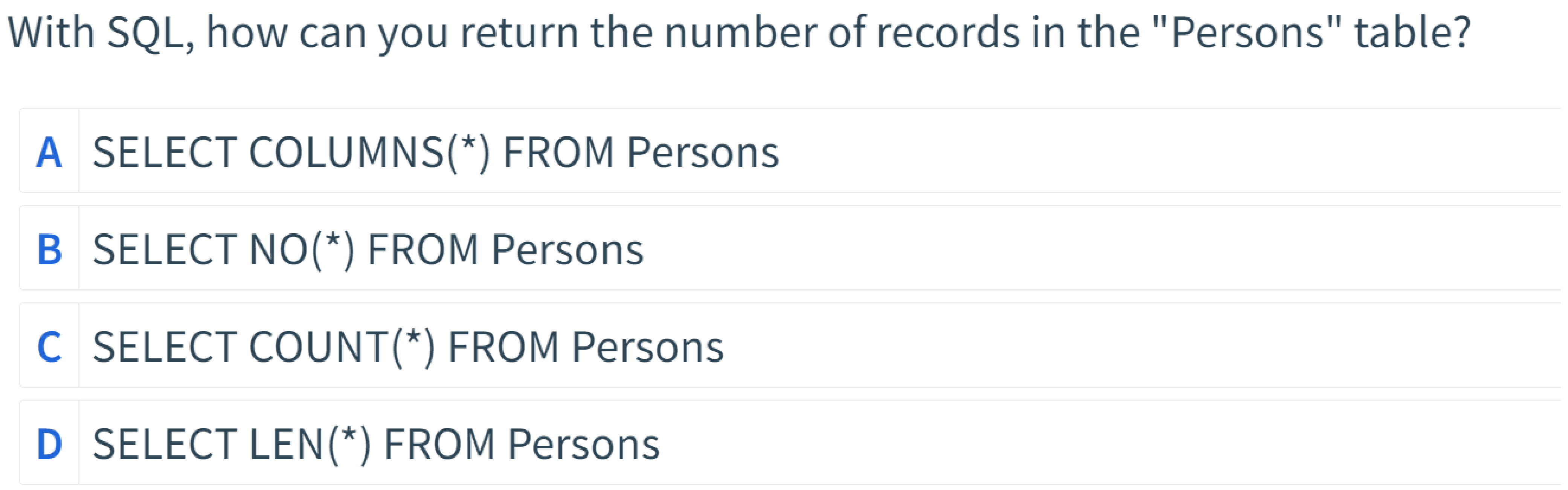
A SELECT * FROM Persons ORDER BY FirstName DESC

B SELECT * FROM Persons ORDER FirstName DESC

C SELECT * FROM Persons SORT 'FirstName' DESC

D SELECT * FROM Persons SORT BY 'FirstName' DESC

Question 4



With SQL, how can you return the number of records in the "Persons" table?

A SELECT COLUMNS(*) FROM Persons

B SELECT NO(*) FROM Persons

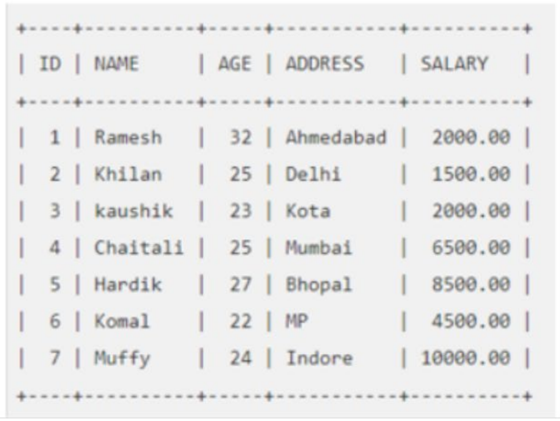
C SELECT COUNT(*) FROM Persons

D SELECT LEN(*) FROM Persons

Question 5

If the following query is submitted in the CUSTOMERS table, how many rows would be in the output?

```
SELECT NAME, SUM(SALARY)
FROM CUSTOMERS
GROUP BY NAME;
```



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

A 0 rows

B 6 rows

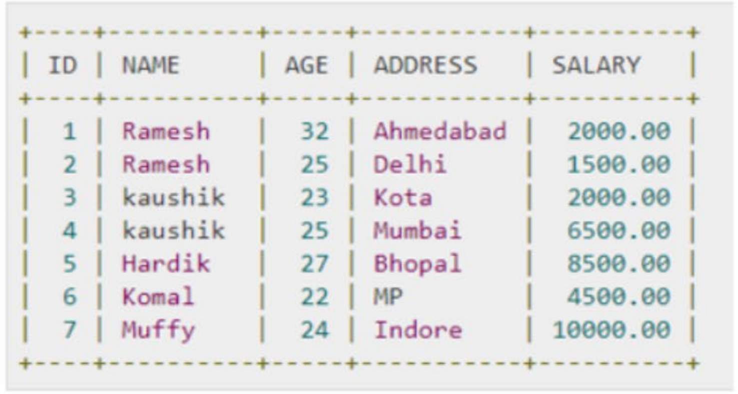
C 7 rows

D 8 rows

Question 6

What about now?

```
SELECT NAME, SUM(SALARY)
FROM CUSTOMERS
GROUP BY NAME;
```



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Ramesh   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | kaushik  |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

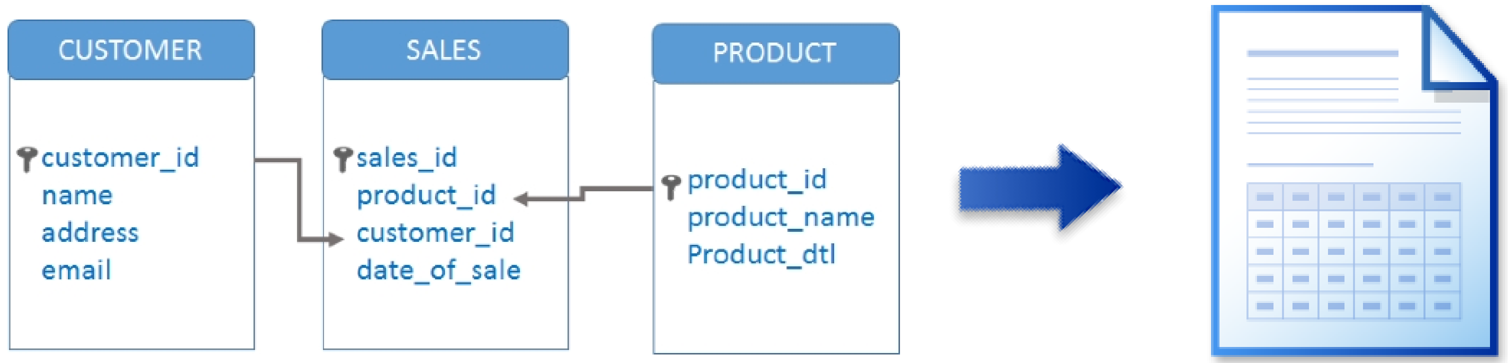
A	0 rows
B	4 rows
C	5 rows
D	7 rows

Adding the new database Practice to DataGrip

SQL Joins

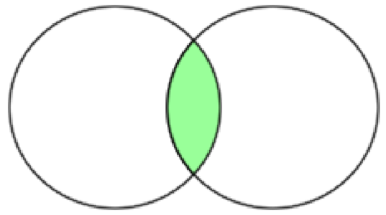
Combining Tables

SQL uses *joins* to combine tables **horizontally**. Requesting a join involves matching data from one row in one table with a corresponding row in a second table. Matching is typically performed on one or more columns in the two tables.

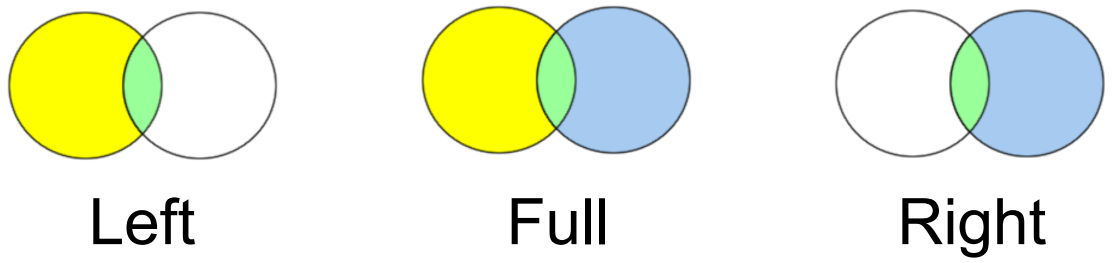


Types of Joins: two types

- *Inner joins* return only matching rows.



- *Outer joins* return all matching rows, plus nonmatching rows from one or both tables.



Cartesian Product

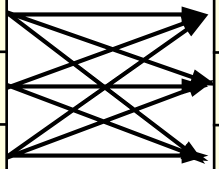
A query that lists multiple tables in the FROM clause without a WHERE clause produces all possible combinations of rows from all tables. This result is called a *Cartesian product*.

```
select *
from customers, transactions;
```

Cartesian Product

customers

ID	Name
101	Smith
104	Jones
102	Blank



transactions

ID	Action	Amount
102	Purchase	$100
103	Return	$52
105	Return	$212

Result Set

Non-matching IDs

```
 ID   Name     ID   Action        Amount
----------------------------------------
101   Smith    102  Purchase        $100
101   Smith    103  Return           $52
101   Smith    105  Return          $212
104   Jones    102  Purchase        $100
104   Jones    103  Return           $52
104   Jones    105  Return          $212
102   Blank    102  Purchase        $100
102   Blank    103  Return           $52
102   Blank    105  Return          $212
```

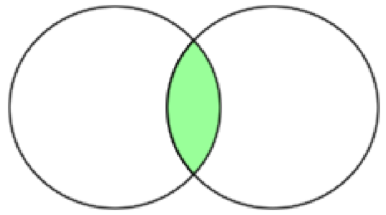
9 rows

The Cartesian Product is rarely what we want to produce…

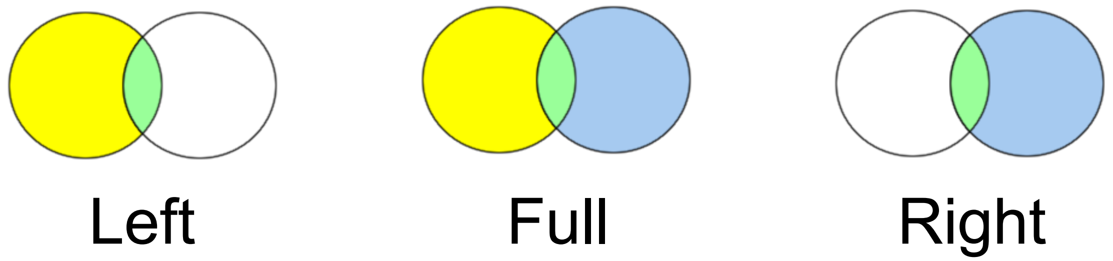
Inner Joins

Types of Joins: two types

- *Inner joins* return only matching rows.

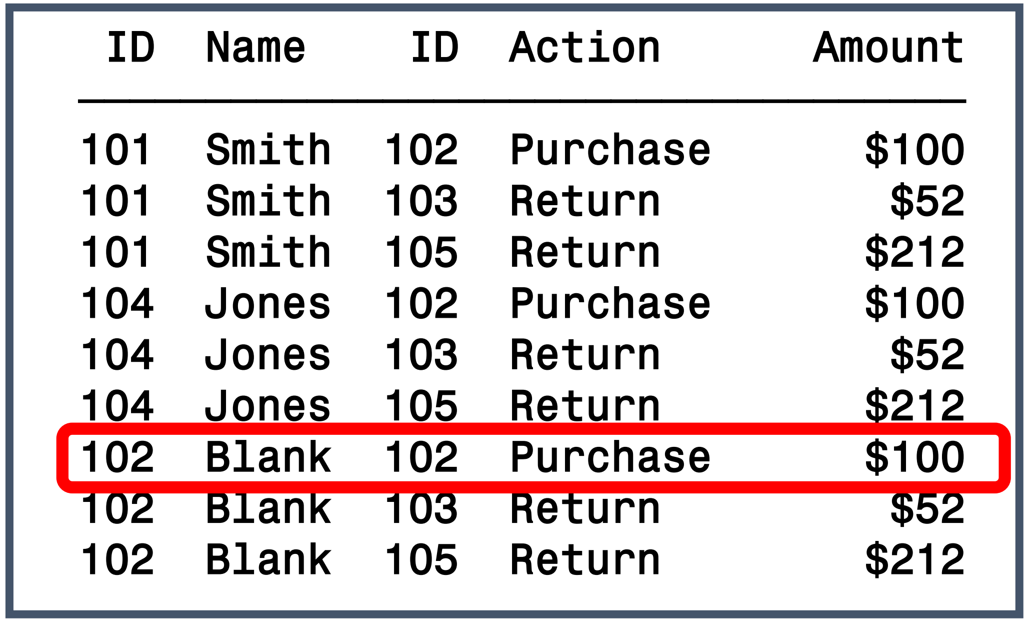


- *Outer joins* return all matching rows, plus nonmatching rows from one or both tables.



Inner Join

Generate a report showing all valid order information:



```
  ID  Name    ID   Action          Amount
 ________________________________________

 101  Smith  102  Purchase          $100
 101  Smith  103  Return             $52
 101  Smith  105  Return            $212
 104  Jones  102  Purchase          $100
 104  Jones  103  Return             $52
 104  Jones  105  Return            $212
 102  Blank  102  Purchase          $100
 102  Blank  103  Return             $52
 102  Blank  105  Return            $212
```

Inner Join

The inner join clause links two (or more) tables by a relationship between two columns.

```
select *
from customers, transactions
where customers.ID=transactions.ID;
```

SELECT *object-item<, …object-item>*
    FROM *table-name, … table-name*
    WHERE *join condition*
            <**AND** *sql-expression*>
    <other clauses>;

Abbreviating the Code

- A *table alias* is a temporary, alternative name for a table. You can make the query easier to read by using table aliases.

```
SELECT alias-1.object-item<, …alias-2.object-item>
       FROM table-name <AS> alias-1,
            table-name <AS> alias-2
       WHERE join-condition(s)
       <other clauses>;
```

- The AS keyword is optional in the table alias syntax.

Abbreviating the Code

```
proc sql;
select c.ID, Name, Action, Amount
   from customers as c, transactions as t
   where c.ID=t.ID;
quit;
```

```
 ID  Name    Action        Amount
─────────────────────────────────
102  Blank   Purchase        $100
```

Alternative Join Syntax

This alternative syntax names the join type and includes an ON clause

```
select c.ID, Name, Action, Amount
   from customers as c
        inner join
        transactions as t
        on c.ID=t.ID;
```

SELECT *object-item* <, …object-item>
    FROM *table-name* <<**AS**> *alias*>
        INNER JOIN
        table-name <<**AS**> *alias*>
    ON *join-condition(s)*
    WHERE *sql-expression*
    <*other clauses*>**;**

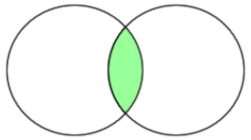
Tables: **`jupiter.employees`**
`jupiter.employee_addresses`
Display: Employee_ID, Gender, Employee_Name

Tables: **`practice.movies`**
`practice.genres`
Display: movie_name, genre name

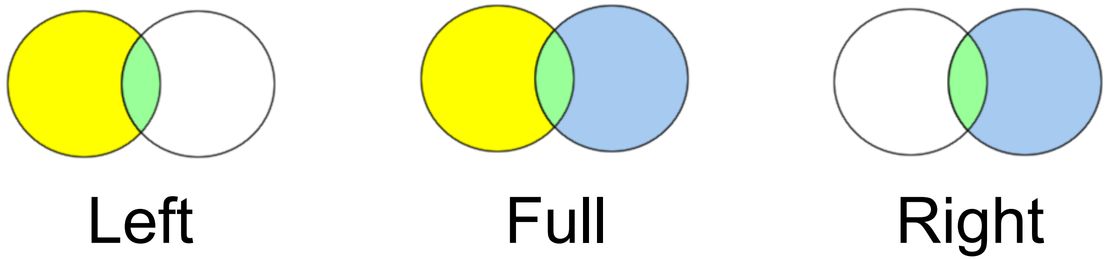
Outer Joins

Outer Joins

- *Inner joins* return only matching rows.

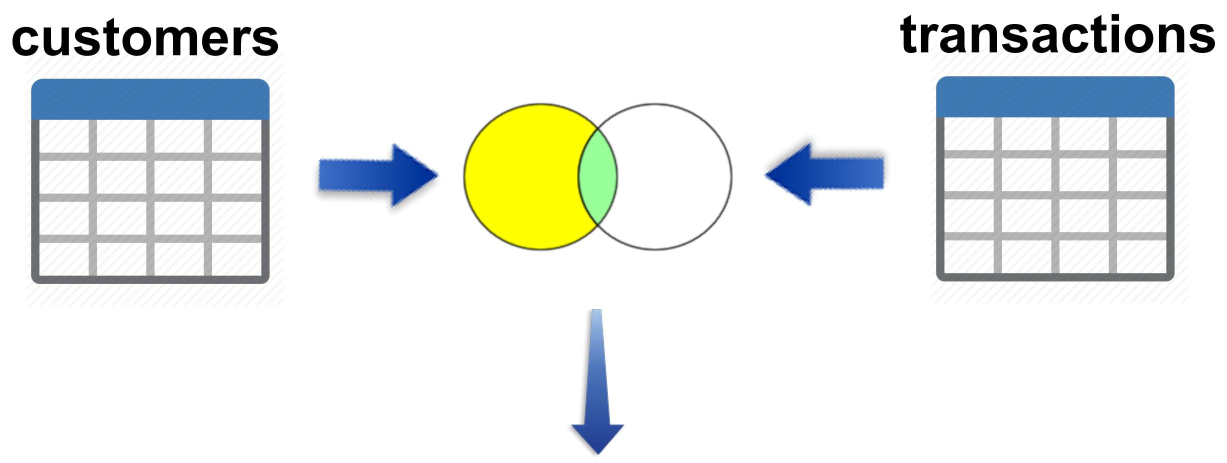


- *Outer joins:* you can retrieve both non-matching and matching rows using an outer join. Many tables can be referenced in outer joins. The tables are processed two tables at a time.



Outer Joins

Generate a report that displays ***all*** customers and any transactions that they have completed.



```
 ID   Name      ID   Action          Amount
-------------------------------------------
101   Smith      .                       .
102   Blank    102   Purchase         $100
104   Jones      .                       .
```

Outer Joins

Outer join syntax is similar to the alternate inner join syntax.

```
select *
   from customers as c
     left join
     transactions as t
   on c.ID=t.ID;
```

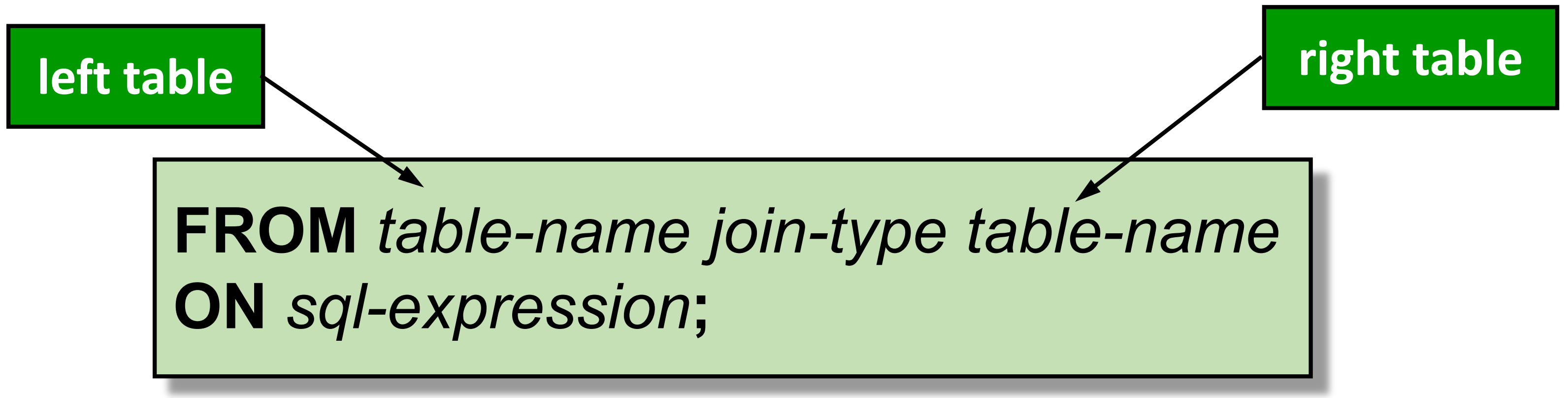
The ON clause specifies the join criteria in outer joins.

SELECT *object-item* <, …object-item>
    FROM *table-name* <<**AS**> *alias*>
        LEFT|**RIGHT**|**FULL JOIN**
        table-name <<**AS**> *alias*>
    ON *join-condition(s)*
    <other clauses>;

Determining Left and Right

Consider the position of the tables in the FROM clause.

- Left joins return all matching and non-matching rows from the ***left*** table and the matching rows from the ***right*** table.
- Right joins return all matching and non-matching rows from the ***right*** table and the matching rows from the ***left*** table.
- Full joins return all matching and non-matching rows from all of the tables.



FROM *table-name join-type table-name*
ON *sql-expression*;

Left Join

customers

ID	Name
101	Smith
104	Jones
102	Blank

transactions

ID	Action	Amount
102	Purchase	$100
103	Return	$52
105	Return	$212

```
select *
   from customers c left join transactions t
   on c.ID = t.ID;
```

```
  ID   Name     ID   Action        Amount
 ----------------------------------------
 101   Smith     .                      .
 102   Blank   102   Purchase        $100
 104   Jones     .                      .
```

Includes all rows from the left table, even if there are no matching rows in the right table.

Right Join

customers

ID	Name
101	Smith
104	Jones
102	Blank

transactions

ID	Action	Amount
102	Purchase	$100
103	Return	$52
105	Return	$212

```
select *
   from customers c right join transactions t
   on c.ID = t.ID;
```

```
  ID  Name      ID  Action          Amount
 ------------------------------------------
 102  Blank    102  Purchase          $100
   .           103  Return             $52
   .           105  Return            $212
```

Includes all rows from the right table, even if there are no matching rows in the left table.

Full Join

customers

ID	Name
101	Smith
104	Jones
102	Blank

transactions

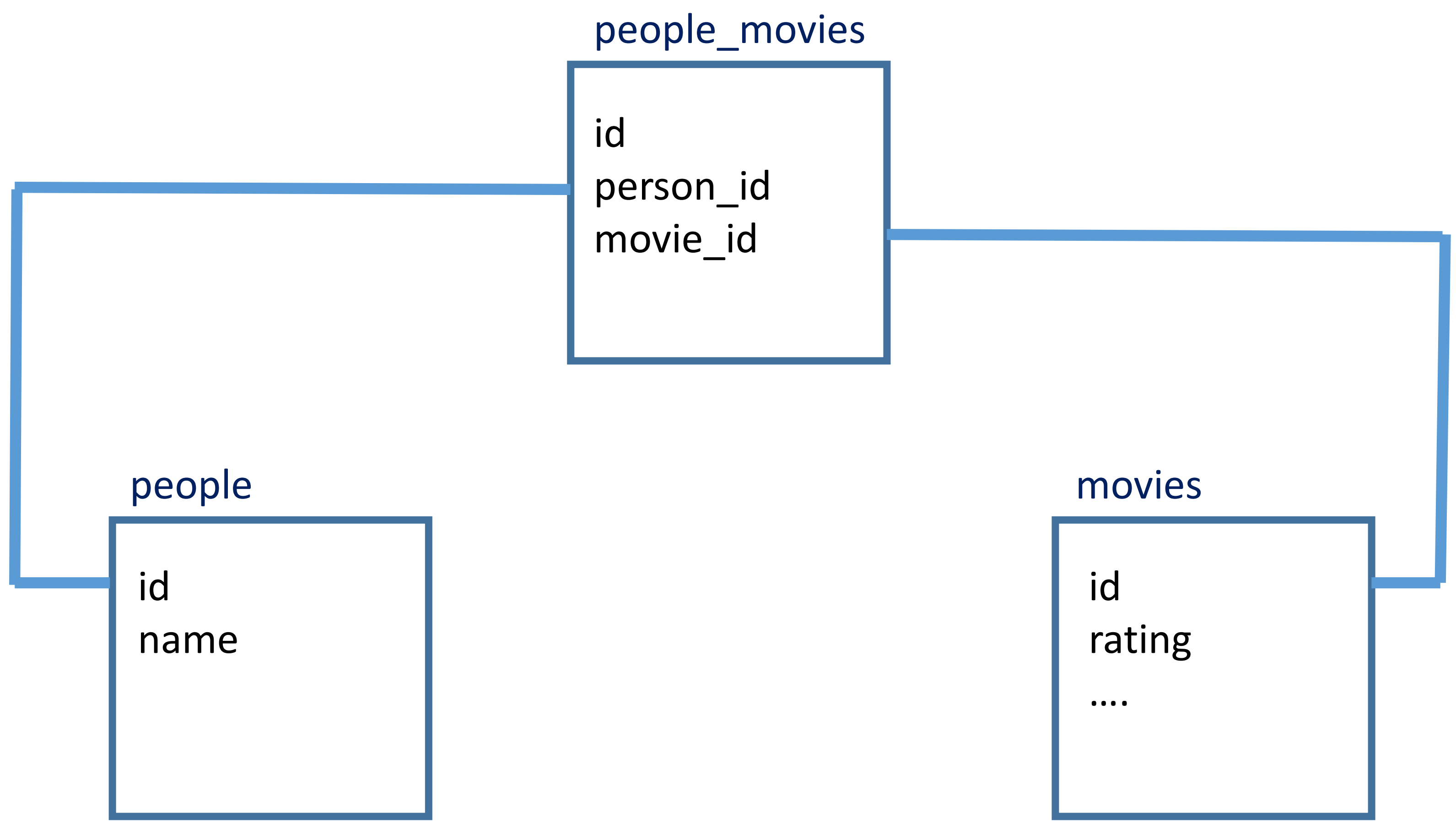
ID	Action	Amount
102	Purchase	$100
103	Return	$52
105	Return	$212

```
select *
   from customers c full join transactions t
   on c.ID = t.ID;
```

```
  ID  Name     ID  Action       Amount
 ____________________________________
 101  Smith     .                    .
 102  Blank   102  Purchase       $100
   .          103  Return          $52
 104  Jones     .                    .
   .          105  Return         $212
```

Includes all rows from both tables, even if there are no matching rows in either table

For the lab…


people_movies
id
person_id
movie_id
people
id
name
movies
id
rating
….