|  |  |  |  |
| --- | --- | --- | --- |
| Indicator | Poor | Average | Good |
| Timeliness   * Maintains assignment   deadline (3) | Assignment not done (0) | One or More than One week late (1-2) | Maintains deadline (3) |
| Completeness and neatness   * Complete all parts of   assignment(3) | N/A | < 80% complete  (1-2) | 100%  complete (3) |
| Originality   * Extent of plagiarism(2) | Copied it from someone  else(0) | At least few questions have been done without copying(1) | Assignment has been  solved  completely without  copying (2) |
| Knowledge   * In depth knowledge of the assignment(2) | Unable to answer 2  questions(0) | Unable to answer 1 question (1) | Able to answer 2 questions (2) |

|  |  |
| --- | --- |
| Timeliness |  |
| Completeness and neatness |  |
| Originality |  |
| Knowledge |  |
| Total |  |

|  |
| --- |
| SE-COMPUTER Roll number : 9913 |
| Experiment no. : 6 Date of Implementation : 12/ 3/ 2024 |
| Aim : To implement Join and complex SQL commands |
| Tool Used : PostgreSQL |
| Related Course outcome : At the end of the course, Students will be able to Use SQL : Standard language of relational database |
| **Rubrics for assessment of Experiment:** |
| **Assessment Marks :** |
| **Total : (Out of 10)** |

|  |  |
| --- | --- |
|  | |
| **Teacher's Sign :** | |
| ***EXPE RIME NT 5*** | Complex SQL commands |
| Aim | To implement complex SQL queries |
| Tools | PostgreSQL |
| Theor y | **Joining Tables**  The FROM clause allows more than 1 table in its list, however simply listing more than one table will *very* rarely produce the expected results. The rows from one table must be correlated with the rows of the others. This correlation is known as *joining*. In the subsequent text, the following 3 example tables are used:  **p Table (parts) s Table (suppliers) sp Table (suppliers & parts)**  An example can best illustrate the rationale behind joins. The following query:  **SELECT \* FROM sp, p**  Produces:  Each row in *sp* is arbitrarily combined with each row in *p*, giving 12 result rows (4 rows in *sp* X 3 rows in *p*.) This is known as a *cartesian product*. |

|  |  |  |
| --- | --- | --- |
| **pno** | **descr** | **color** |
| P1 | Widget | Blue |
| P2 | Widget | Red |
| P3 | Dongle | Green |

|  |  |  |
| --- | --- | --- |
| **sno** | **name** | **city** |
| S1 | Pierre | Paris |
| S2 | John | London |
| S3 | Mario | Rome |

|  |  |  |
| --- | --- | --- |
| **sno** | **pno** | **qty** |
| S1 | P1 | NULL |
| S2 | P1 | 200 |
| S3 | P1 | 1000 |
| S3 | P2 | 200 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sno** | **pno** | **qty** | **pno** | **descr** | **color** |
| S1 | P1 | NULL | P1 | Widget | Blue |
| S1 | P1 | NULL | P2 | Widget | Red |
| S1 | P1 | NULL | P3 | Dongle | Green |
| S2 | P1 | 200 | P1 | Widget | Blue |
| S2 | P1 | 200 | P2 | Widget | Red |
| S2 | P1 | 200 | P3 | Dongle | Green |
| S3 | P1 | 1000 | P1 | Widget | Blue |
| S3 | P1 | 1000 | P2 | Widget | Red |
| S3 | P1 | 1000 | P3 | Dongle | Green |
| S3 | P2 | 200 | P1 | Widget | Blue |
| S3 | P2 | 200 | P2 | Widget | Red |
| S3 | P2 | 200 | P3 | Dongle | Green |

A more usable query would correlate the rows from *sp* with rows from *p*, for instance

matching on the common column -- *pno*:

**SELECT \***

**FROM sp, p**

**WHERE sp.pno = p.pno**

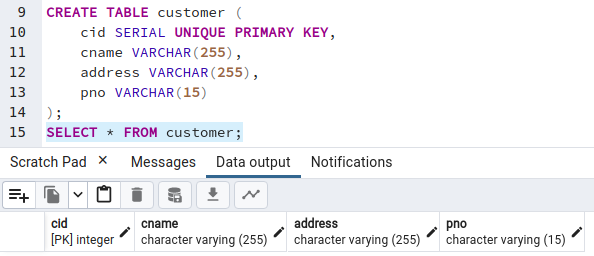
This produces:

More information refer this

http[s://w](http://www.tutorialspoint.com/sql/sql-using-joins.htm)ww.[tutorialspoint.com/sql/sql-using-joins.htm](http://www.tutorialspoint.com/sql/sql-using-joins.htm)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sno** | **pno** | **qty** | **pno** | **descr** | **color** |
| S1 | P1 | NULL | P1 | Widget | Blue |
| S2 | P1 | 200 | P1 | Widget | Blue |
| S3 | P1 | 1000 | P1 | Widget | Blue |
| S3 | P2 | 200 | P2 | Widget | Red |

# Proce dure



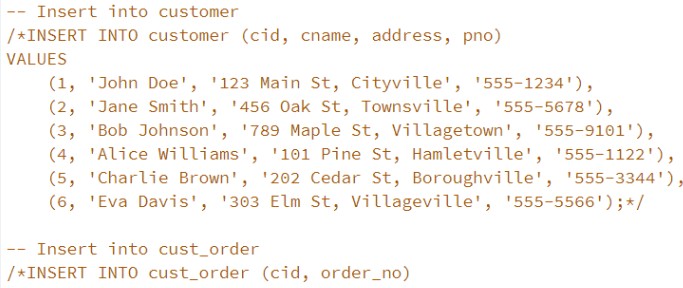
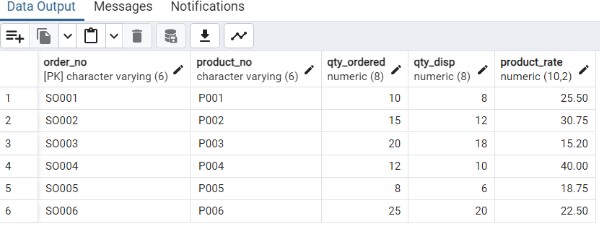
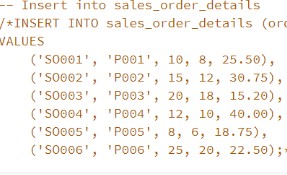
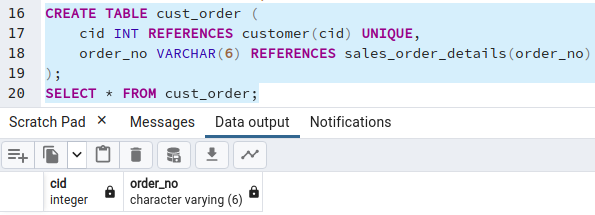
1. Create following table:

Table name : sales\_order\_details

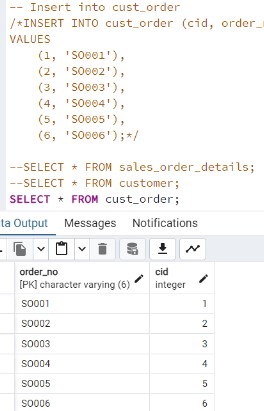
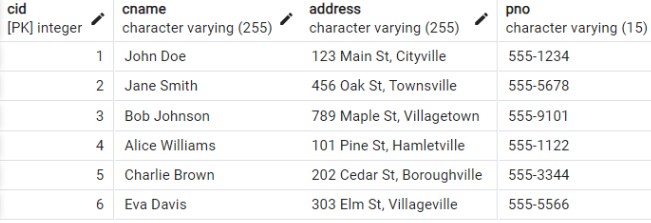
|  |  |  |
| --- | --- | --- |
| Column Name | Data type | Size |
| order\_no | varchar | 6 |
| Product\_no | varchar | 6 |
| Qty\_ordered | numeric | 8 |
| Qty\_disp | numeric | 8 |
| Product\_rate | numeric | 10,2 |

# Create table- customer(cid, cname, address, pno)

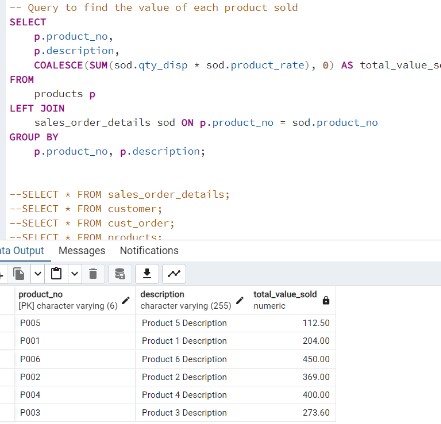
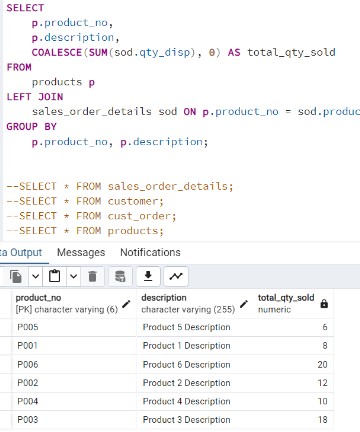
Create table- cust\_order(cid foreign key, order\_no foreign key)



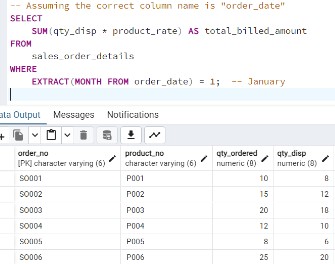
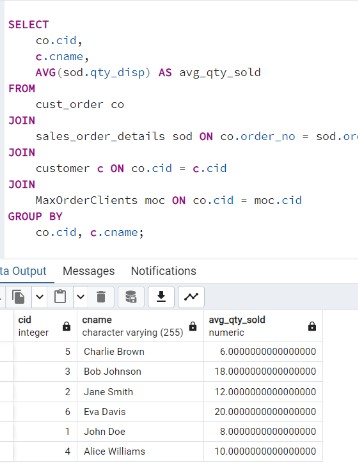
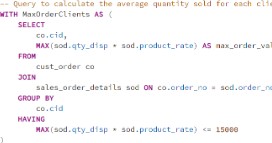
1. Insert 5-6 records in table in each tables.



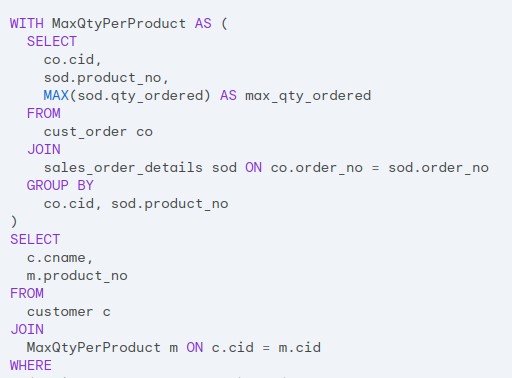
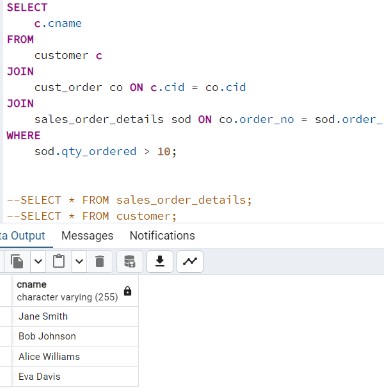
3. Print the description and total qty sold for each product



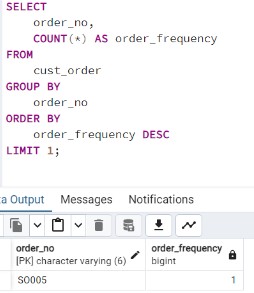
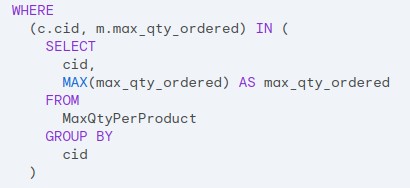
1. Find the value of each product sold
2. Calculate the average quantity sold for each client that has a maximum order value of 15000



1. Find out the sum total of all the billed orders for the month of January
2. Find out the name of customers who have given the order of more than 10 qty.



1. Find out the customer names with product no with maximum qty ordered.



9. Find out most frequent orders

**Post Lab Quest ions:**

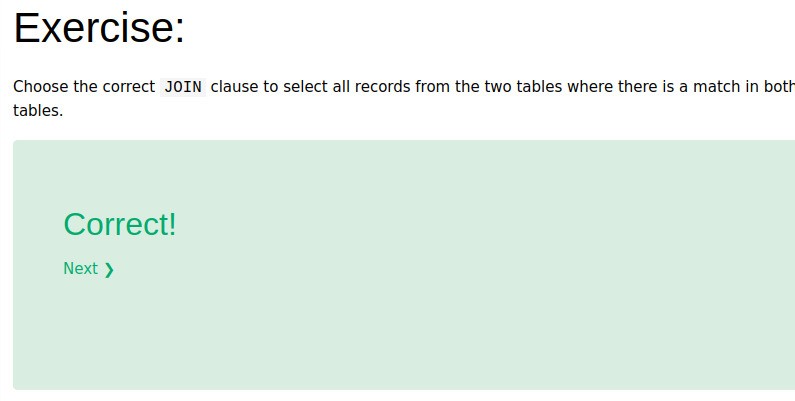
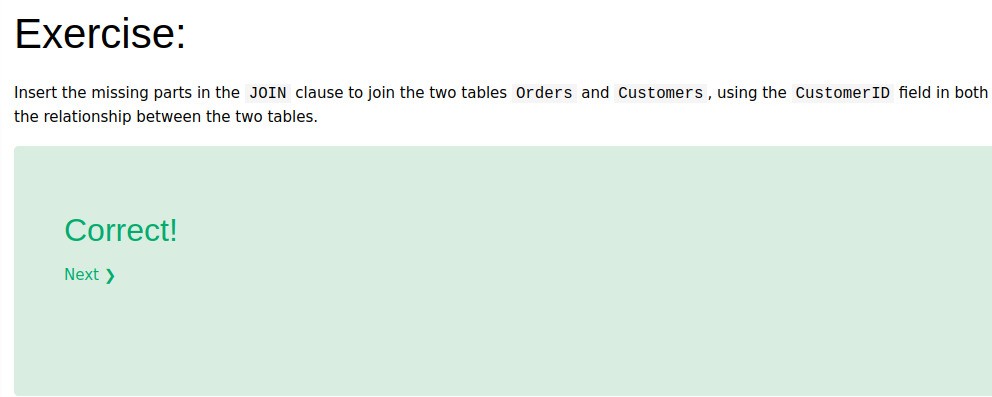
1. What is the difference between inner Join and outer Join.

⇒ The main difference between inner join and outer join in SQL is that an inner join returns only the rows with matching values in both tables, while an outer join returns all the rows from the database tables, including those that do not have a match in the other table. There are three types of outer joins: left outer join, right outer join, and full outer join. An inner join is a simple join that provides the result directly, while an outer join is a complex join that requires additional syntax to specify the type of join. Outer joins are generally faster than inner joins because they are less restrictive and do not require precise matches.

1. Give one example for equi\_join and non equi\_join.

⇒

* + In an Equi Join, the join operation is based on an equality condition using the equals sign (=).For instance, consider two tables: "state" and "city." The "state" table contains State\_ID and State\_Name columns, while the "city" table contains City\_ID and City\_Name columns. An Equi Join can be used to map cities with the states they belong to based on a common column.



* + In a Non Equi Join, the join condition involves comparison operators other than the equals sign, such as >, <, >=, <=. For example, consider two tables: "orders" and "customer." To retrieve order numbers and order amounts from the "orders" table and customer names and working areas from the "customer" table where the order amount matches any opening amount in the customer table, a Non Equi Join can be used.

1. complete online exercise and add screen shots <https://www.w3schools.com/sql/exercise.asp?filename=exercise_join1>

