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:

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 100% complete	
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)
KnowledgeIn depth knowledge of the assignment(2)	Unable to answer 2 questions(0)	Unable to answer 1 question (1)	Able to answer 2 questions (2)

Assessment Marks:

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Total: (Out of 10)

Teach	er's Sign :
EXPE	Complex SQL commands
RIME	
NT 5	
Aim	To implement complex SQL queries
Tools	PostgreSQL
Theor	
У	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)

pno	descr	color
P1	Widget	Blue
P2	Widget	Red
Р3	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

An example can best illustrate the rationale behind joins. The following query:

SELECT * FROM sp, p

Produces:

CJ.					
sno	pno	qty	pno	descr	color
S1	P1	NULL	P1	Widget	Blue
S1	P1	NULL	P2	Widget	Red
S1	P1	NULL	Р3	Dongle	Green
S2	P1	200	P1	Widget	Blue
S2	P1	200	P2	Widget	Red
S2	P1	200	Р3	Dongle	Green
S3	P1	1000	P1	Widget	Blue
S3	P1	1000	P2	Widget	Red
S3	P1	1000	Р3	Dongle	Green
S3	P2	200	P1	Widget	Blue
S3	P2	200	P2	Widget	Red
S3	P2	200	Р3	Dongle	Green

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*.

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:

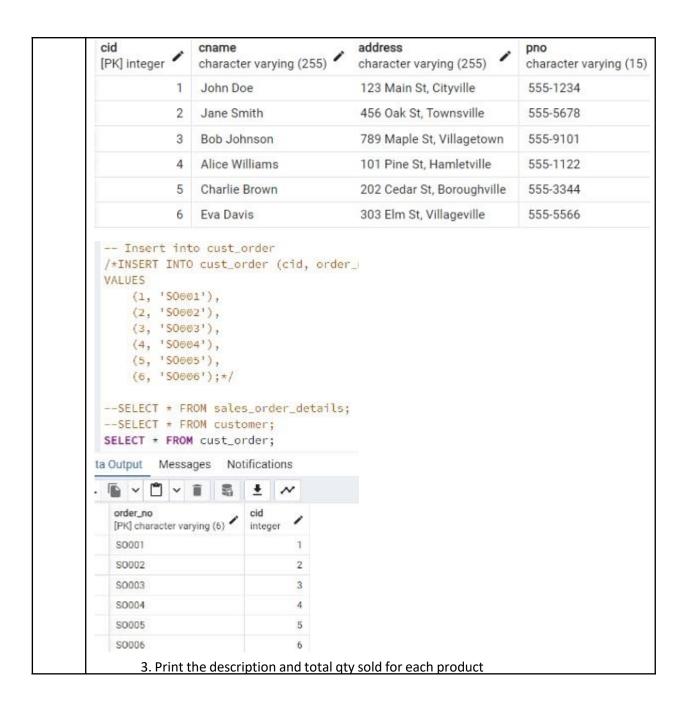
sno	pno	qty	pno	descr	color
S1	P1	NULL	P1	Widget	Blue
S2	P1	200	P1	Widget	Blue
S3	P1	1000	P1	Widget	Blue
S3	P2	200	P2	Widget	Red

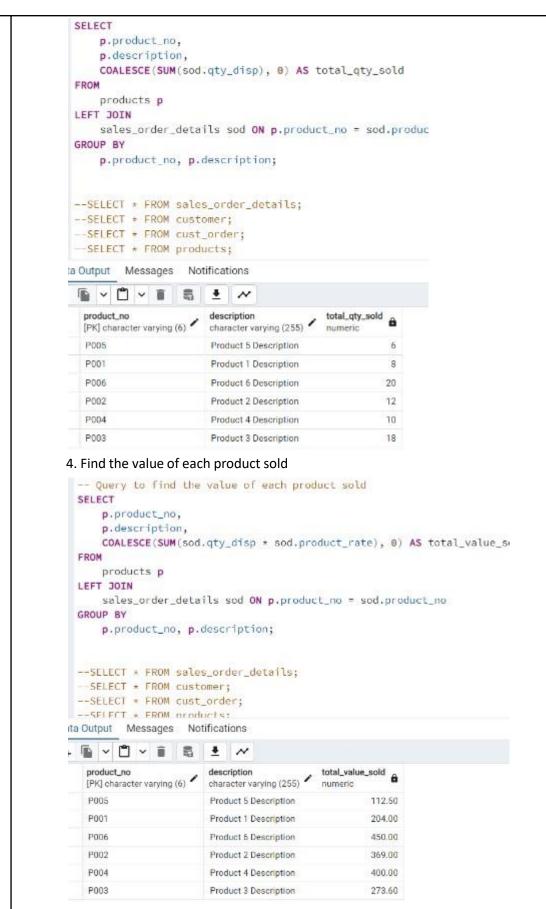
More information refer this

https://www.tutorialspoint.com/sql/sql-using-joins.htm

1. Create following table: Proce Table name: sales_order_details dure Column Data Size Name type 6 order_no varchar Product no varchar 6 Qty_ordered numeric 8 8 Qty_disp numeric 10,2 Product_rate numeric Create table- customer(cid, cname, address, pno) Create table- cust order(cid foreign key, order no foreign key) CREATE TABLE sales_order_details (1 2 order_no VARCHAR(6) REFERENCES salesorder(order_no), 3 Product_no VARCHAR(6), 4 Qty_ordered NUMERIC(8), 5 Qty_disp NUMERIC(8), 6 Product_rate NUMERIC(10,2) 7); 8 SELECT * FROM sales_order_details; 9 Scratch Pad X Messages Data output Notifications product_no
character varying (6) a qty_ordered numeric (8) a qty_disp numeric (8) product_no product_rate numeric (10,2) 9 CREATE TABLE customer (10 cid SERIAL UNIQUE PRIMARY KEY, 11 cname VARCHAR(255), 12 address VARCHAR(255), 13 pno VARCHAR(15) 14 SELECT * FROM customer; Scratch Pad X Notifications Messages Data output address character varying (255) character varying (255) character varying (15)

```
CREATE TABLE cust_order (
 16
 17
          cid INT REFERENCES customer(cid) UNIQUE,
          order_no VARCHAR(6) REFERENCES sales_order_details(order_no)
 18
 19
 20
     SELECT * FROM cust_order;
 Scratch Pad X
                            Data output
                                         Notifications
                 Messages
=+
                 order_no
      integer
                 character varying (6)
      2. Insert 5-6 records in table in each tables.
-- Insert into sales_order_details
/*INSERT INTO sales_order_details (ord
VALUES
    ('S0001', 'P001', 10, 8, 25.50),
    ('50002', 'P002', 15, 12, 30.75),
    ('S0003', 'P003', 20, 18, 15.20),
    ('S0004', 'P004', 12, 10, 40.00),
   ('S0005', 'P005', 8, 6, 18.75),
   ('S0006', 'P006', 25, 20, 22.50);
 Data Output Messages Notifications
=+ • ~ • ~ •
                          <u>+</u> ~
                          product_no
                                           qty_ordered
numeric (8)
                                                                  product_rate
                                                      qty_disp
      order no
                                                      numeric (8)
                                                                  numeric (10,2)
                          character varying (6)
      [PK] character varying (6)
      S0001
                          P001
 1
                                                   10
                                                               8
                                                                         25.50
2
      S0002
                          P002
                                                   15
                                                              12
                                                                         30.75
3
      S0003
                          P003
                                                                         15.20
                                                   20
                                                              18
 4
      S0004
                          P004
                                                                         40.00
                                                   12
                                                              10
                                                                         18.75
5
      S0005
                          P005
                                                   8
                                                               6
      S0006
                          P006
                                                                         22.50
                                                   25
                                                              20
-- Insert into customer
/*INSERT INTO customer (cid, cname, address, pno)
VALUES
     (1, 'John Doe', '123 Main St, Cityville', '555-1234'),
     (2, 'Jane Smith', '456 Oak St, Townsville', '555-5678'),
     (3, 'Bob Johnson', '789 Maple St, Villagetown', '555-9101'),
     (4, 'Alice Williams', '101 Pine St, Hamletville', '555-1122'),
     (5, 'Charlie Brown', '202 Cedar St, Boroughville', '555-3344'),
     (6, 'Eva Davis', '303 Elm St, Villageville', '555-5566');*/
-- Insert into cust_order
/*INSERT INTO cust_order (cid, order_no)
```





5. Calculate the average quantity sold for each client that has a maximum order value of 15000

```
ry to calculate the average quantity sold for each clim
WITH MaxOrderClients AS (
   SELECT
       co.cid.
       MAX(sod.qty_disp * sod.product_rate) AS max_order_val
       cust_order co
   JOIN
       sales_order_details sod ON co.order_no = sod.order_no
   GROUP BY
       co.cid
   HAVING
       MAX(sod.qty_disp = sod.product_rate) <= 15688
  SELECT
       co.cid,
       c.cname,
       AVG(sod.qty_disp) AS avg_qty_sold
  FROM
       cust_order co
  JOIN
       sales_order_details sod ON co.order_no = sod.or
 JOIN
       customer c ON co.cid = c.cid
  JOIN
       MaxOrderClients moc ON co.cid = moc.cid
 GROUP BY
       co.cid, c.cname;
ta Output
            Messages
                        Notifications
   cid
                cname
                                         avg_qty_sold
                                                            8
                character varying (255)
   integer
                                         numeric
             5 Charlie Brown
                                           6.00000000000000000
                Bob Johnson
                                          18.00000000000000000
                                          12.000000000000000000
             2 Jane Smith
             6 Eva Davis
                                          20.000000000000000000
                John Doe
                                           8.00000000000000000
            4 Alice Williams
                                          10.000000000000000000
6. Find out the sum total of all the billed orders for the month of January
    Assuming the correct column name is "order_date"
  SELECT
      SUM(qty_disp * product_rate) AS total_billed_amount
      sales_order_details
  WHERE
      EXTRACT(MONTH FROM order_date) = 1; -- January
ata Output Messages Notifications
  島
                        * W
   order no
                        product_no
character varying (6)
                                          qty_ordered /
numeric (8)
                                                      atv_disp
                                                      numeric (8)
   [PK] character varying (6)
                                                   10
    S0001
                        P002
                                                   15
                                                              12
   50002
    50003
                        P003
                                                   20
                                                              18
    50004
                        P004
                                                   12
                                                              10
```

80005

50006

P005

P006

6

25

7. Find out the name of customers who have given the order of more than 10 qty.

```
SELECT
     c.cname
 FROM
     customer c
 JOIN
     cust_order co ON c.cid = co.cid
     sales_order_details sod ON co.order_no = sod.order_
 WHERE
     sod.qty_ordered > 10;
 -- SELECT * FROM sales_order_details;
 -- SELECT * FROM customer;
a Output Messages Notifications
  cname
  character varying (255)
  Jane Smith
   Bob Johnson
   Alice Williams
   Eva Davis
8. Find out the customer names with product no with maximum qty ordered.
 WITH MaxQtyPerProduct AS (
    SELECT
      co.cid,
      sod.product no,
      MAX(sod.qty_ordered) AS max_qty_ordered
    FROM
      cust_order co
    JOIN
      sales_order_details sod ON co.order_no = sod.order_no
    GROUP BY
      co.cid, sod.product_no
  )
 SELECT
    c.cname,
    m.product_no
 FROM
    customer c
  JOIN
```

MaxQtyPerProduct m ON c.cid = m.cid

```
WHERE
    (c.cid, m.max_qty_ordered) IN (
       SELECT
         cid,
         MAX(max_qty_ordered) AS max_qty_ordered
       FROM
         MaxQtyPerProduct
       GROUP BY
         cid
                    product_no
cname
character varying (255)
                    character va
                    P006
 Eva Davis
 Charlie Brown
                    P005
 Bob Johnson
                    P003
 Alice Williams
                    P004
 Jane Smith
                    P002
 John Doe
                    P001
9. Find out most frequent orders
 SELECT
     order_no,
     COUNT(*) AS order_frequency
 FROM
     cust_order
 GROUP BY
     order_no
 ORDER BY
     order_frequency DESC
 LIMIT 1;
a Output
         Messages
                   Notifications
  order_frequency
  order_no
  [PK] character varying (6)
                       bigint
  S0005
```

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.
- 2. Give one example for equi_join and non equi_join.

 \Rightarrow

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

Exercise:

Insert the missing parts in the JOIN clause to join the two tables Orders and Customers, using the CustomerID field in both the relationship between the two tables.

Correct!

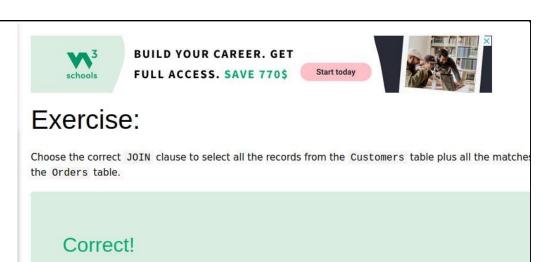
Next >

Exercise:

Choose the correct JOIN clause to select all records from the two tables where there is a match in both tables.

Correct!

Next >



Next >