IT214 Lab 5 Report
Keyur Govrani
202101498

Contents

1 Relational algebraic expressions for all queries

Lab - 5

Page No.
Date

Relational algebraic expression of queries

① $r1 \leftarrow \mathcal{F}_{sum(qty * rate) \rightarrow sales}(Invoicedetails)$
$result \leftarrow \Pi_{sales}(\sigma_{itemcode = 1103}(r1))$

② $\Pi_{invdate, sales}(invdate\ \mathcal{F}_{sum(qty * rate) \rightarrow sales}(Invoicedetails * Invoice))$

③ $\Pi_{itemcode, total\_qty}(itemcode\ \mathcal{F}_{sum(qty) \rightarrow total\_qty}(Invoicedetails))$

④ $r1 \leftarrow itemcode\ \mathcal{F}_{sum(qty) \rightarrow total}(Invoicedetails)$
$\Pi_{code, name, category}(r1.\cancel{itemcode} \bowtie_{r1.itemcode = code} Items)$

⑤ $r1 \leftarrow invno\ \mathcal{F}_{sum(qty * rate) \rightarrow total}(Invoicedetails)$
$result \leftarrow \Pi_{custid}(customer \bowtie_{custid = customerid} Invoice * r_1)$

⑥ $r_1 \leftarrow invno\ \mathcal{F}_{sum(qty * (rate - averagepurchaseprice)) \rightarrow total\_profit}(Invoicedetails$

$\bowtie_{itemcode = code} Items)$
$result \leftarrow \Pi_{custid}(customer \bowtie_{custid = customerid} Invoice * r_1)$

⑦ $r1 \leftarrow year, itemcode\ \mathcal{F}_{sum(qty) \rightarrow total}\ \Pi_{year(invdate) \rightarrow year}($

$Invoicedetails * Invoice)$

$r2 \leftarrow year\ \mathcal{F}_{max(total) \rightarrow max\_qty}(r1)$

$result \leftarrow \Pi_{r1.year, itemcode, total \rightarrow max\_qty}($

$r1 \bowtie_{r1.total = r2.max\_qty\ AND\ r1.year = r2.year} r2)$

(8) $r_1 \leftarrow$ acadyear, semester, instructorid $\mathcal{F}_{COUNT(courseno)}$ (
Instructor * Offers)
result $\leftarrow \Pi$ instructorid (σ count(courseno) > 1 (r_1))

(9) instructor. instructorid, instructorname $\mathcal{F}$ count(courseno) $\rightarrow$ course_count
(instructor $\underset{\text{instructor. instructorid = offers. instructorid}}{\text{LEFT} \bowtie}$ offers)

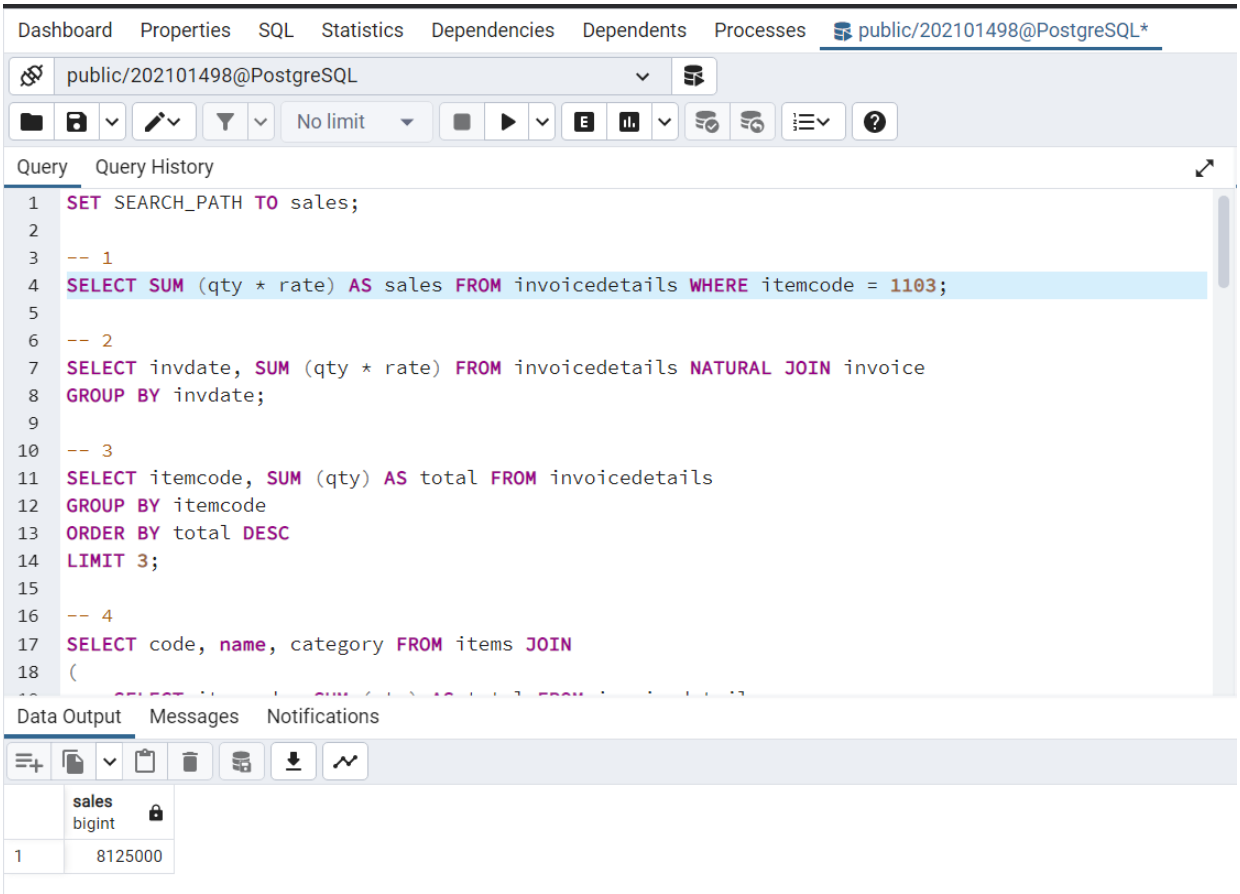
(10) $r_1 \leftarrow$ studentid, name $\mathcal{F}$ sum(credit) $\rightarrow$ total_credits (
student * registers * course)
result $\leftarrow \Pi$ studentid, name, total_credits (σ progid = '02' and batch = '2008'
and semester = 'Autumn' and acadyear = '2008' (r_1))

(11) $r_1 \leftarrow$ studenid, name $\mathcal{F}$ count(grade) (student * registers)
result $\leftarrow \Pi$ studentid, name (σ count(grade) > 2 and semester = 'Autumn'
and acadyear = '2008' and grade = 'F')

2 SQL statements for all queries

1. Compute total sales of a given item (say item code=1103).

```
SELECT SUM (qty * rate) AS sales FROM invoicedetails
WHERE itemcode = 1103;
```



2. What is sale for a given date?

```sql
SELECT invdate, SUM (qty * rate) AS sales FROM invoicedetails
NATURAL JOIN invoice
GROUP BY invdate;
```

Dashboard Properties SQL Statistics Dependencies Dependents Processes public/202101498@PostgreSQL*

public/202101498@PostgreSQL

No limit

Query Query History

```sql
-- 2
SELECT invdate, SUM (qty * rate) AS sales FROM invoicedetails NATURAL JOIN invoice
GROUP BY invdate;

-- 3
SELECT itemcode, SUM (qty) AS total FROM invoicedetails
GROUP BY itemcode
ORDER BY total DESC
LIMIT 3;

-- 4
SELECT code, name, category FROM items JOIN
(
    SELECT itemcode, SUM (qty) AS total FROM invoicedetails
    GROUP BY itemcode
    ORDER BY total DESC
    LIMIT 3
) AS temp ON temp.itemcode = code;
```

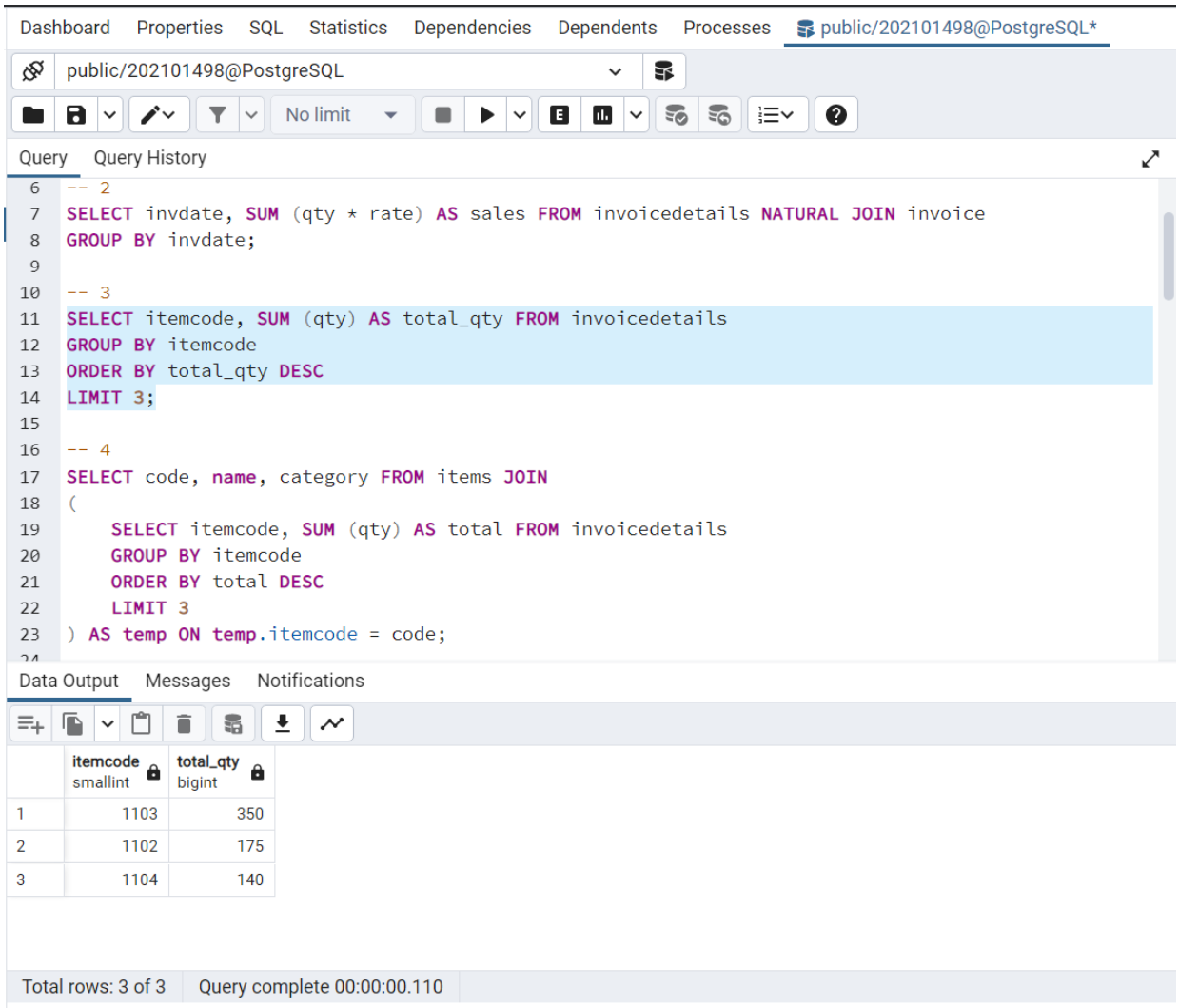
Data Output Messages Notifications

	invdate date	sales bigint
1	2011-08-21	1362500
2	2011-07-26	2820000
3	2011-07-05	1125000
4	2011-08-23	5680000
5	2010-06-30	2880000

Total rows: 5 of 5 Query complete 00:00:00.118

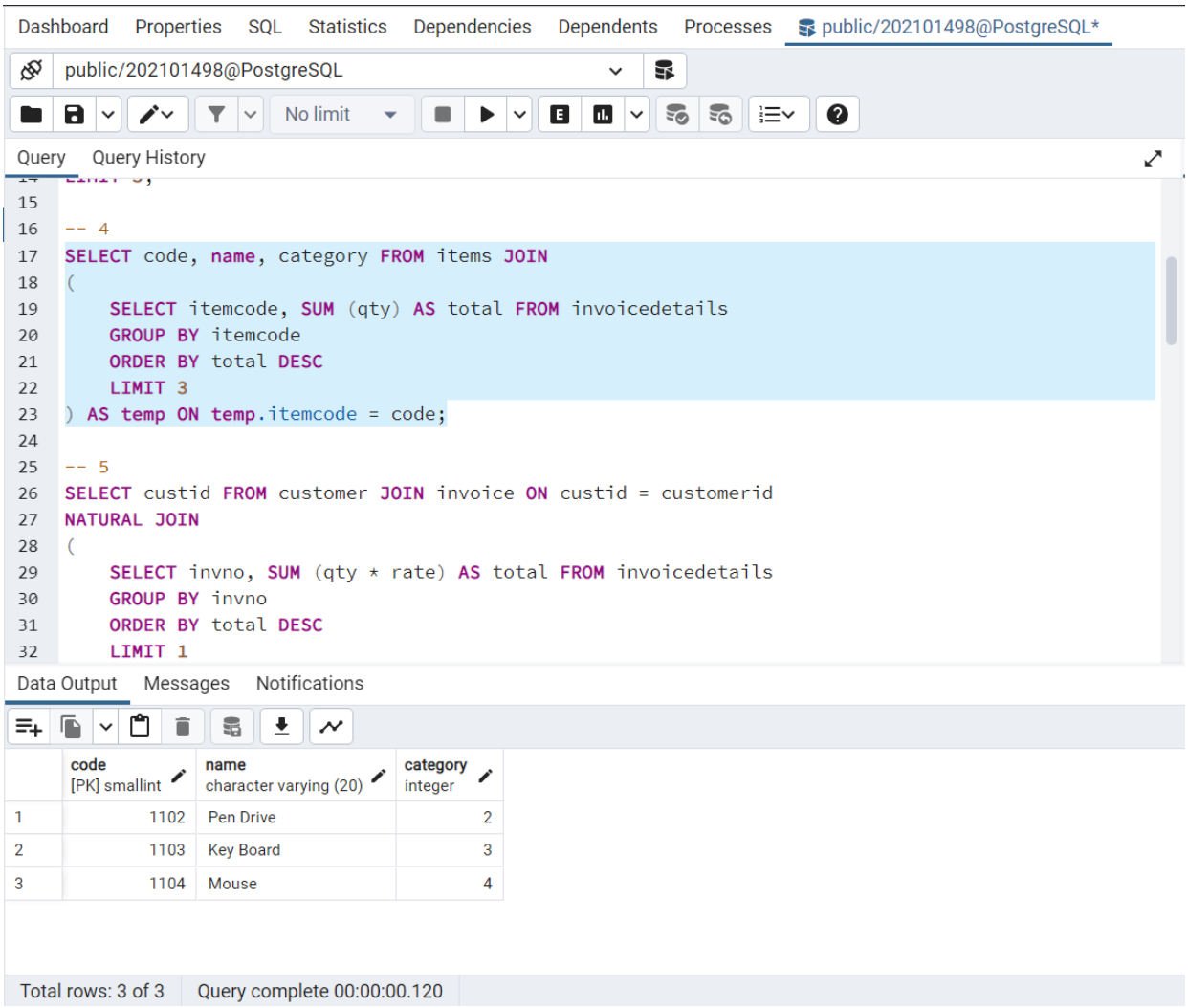
3. List item codes of top 3 most sold item based on quantity

```sql
SELECT itemcode, SUM (qty) AS total_qty FROM invoicedetails
GROUP BY itemcode
ORDER BY total_qty DESC
LIMIT 3;
```



4. List (item codes, item name, category) of top 3 most sold item based on quantity

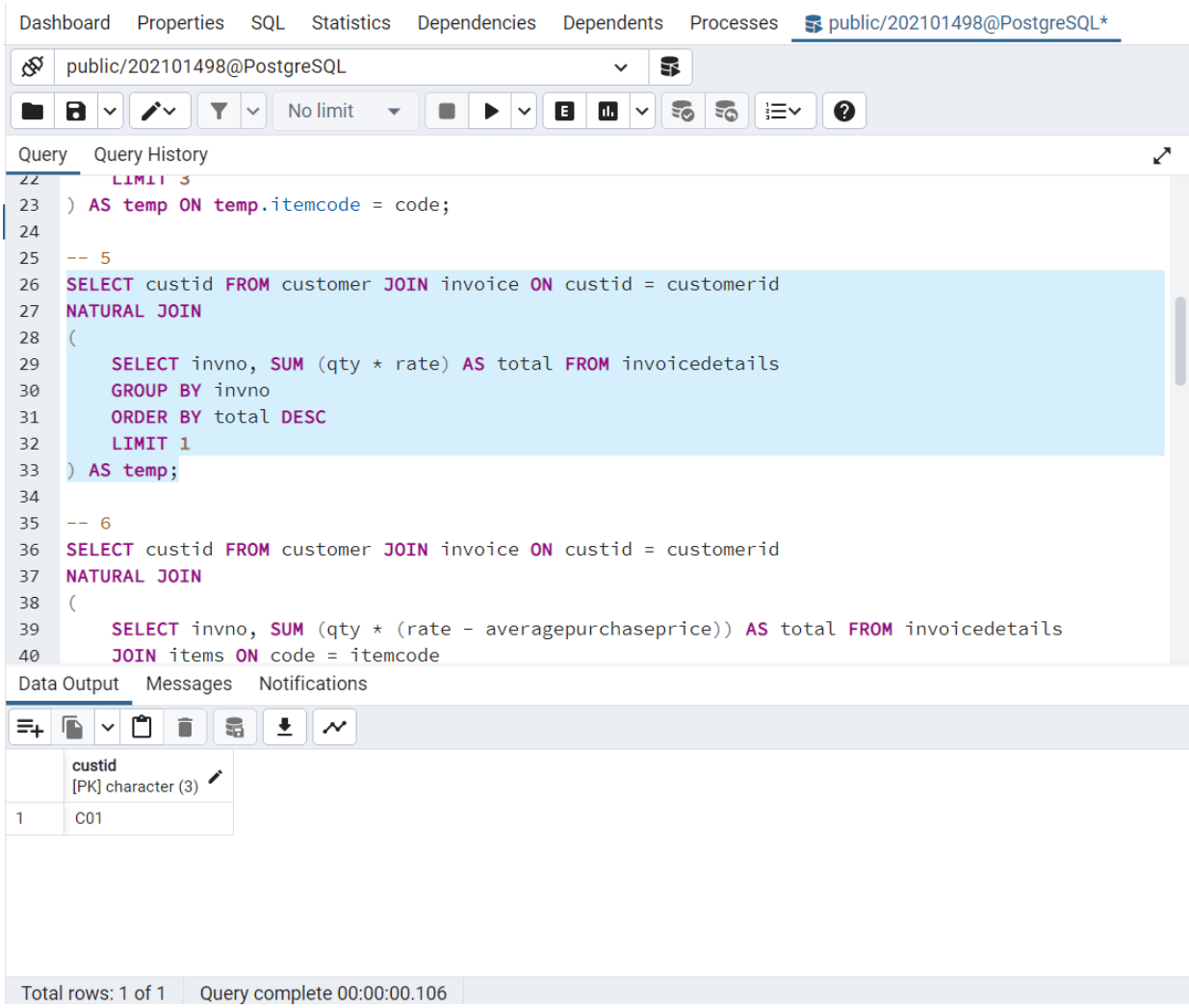
```sql
SELECT code, name, category FROM items JOIN
(
    SELECT itemcode, SUM (qty) AS total FROM invoicedetails
    GROUP BY itemcode
    ORDER BY total DESC
    LIMIT 3
) AS temp ON temp.itemcode = code;
```



	code [PK] smallint	name character varying (20)	category integer
1	1102	Pen Drive	2
2	1103	Key Board	3
3	1104	Mouse	4

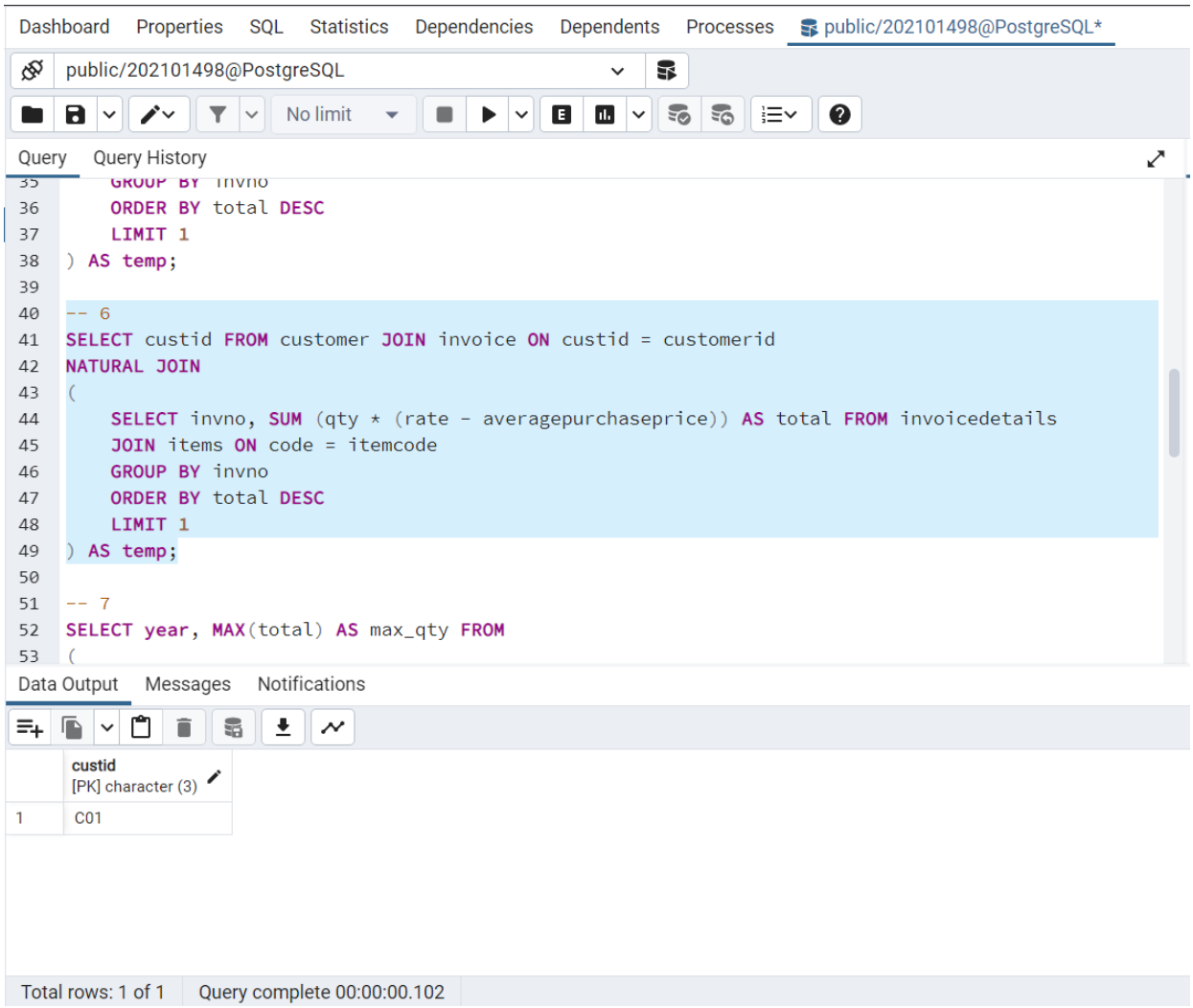
5. Most valuable customer (customer id) in terms of purchase values. Customer that sums of maximum sale amount.

```sql
SELECT custid FROM customer JOIN invoice ON custid = customerid
NATURAL JOIN
(
    SELECT invno, SUM (qty * rate) AS total FROM invoicedetails
    GROUP BY invno
    ORDER BY total DESC
    LIMIT 1
) AS temp;
```



6. Most valuable customer (customer id) in terms profit to the company. Assume that profit on an item sale can be computed by formula:
   Rate (from invoicedetails relation) - AveragePurchasePrice

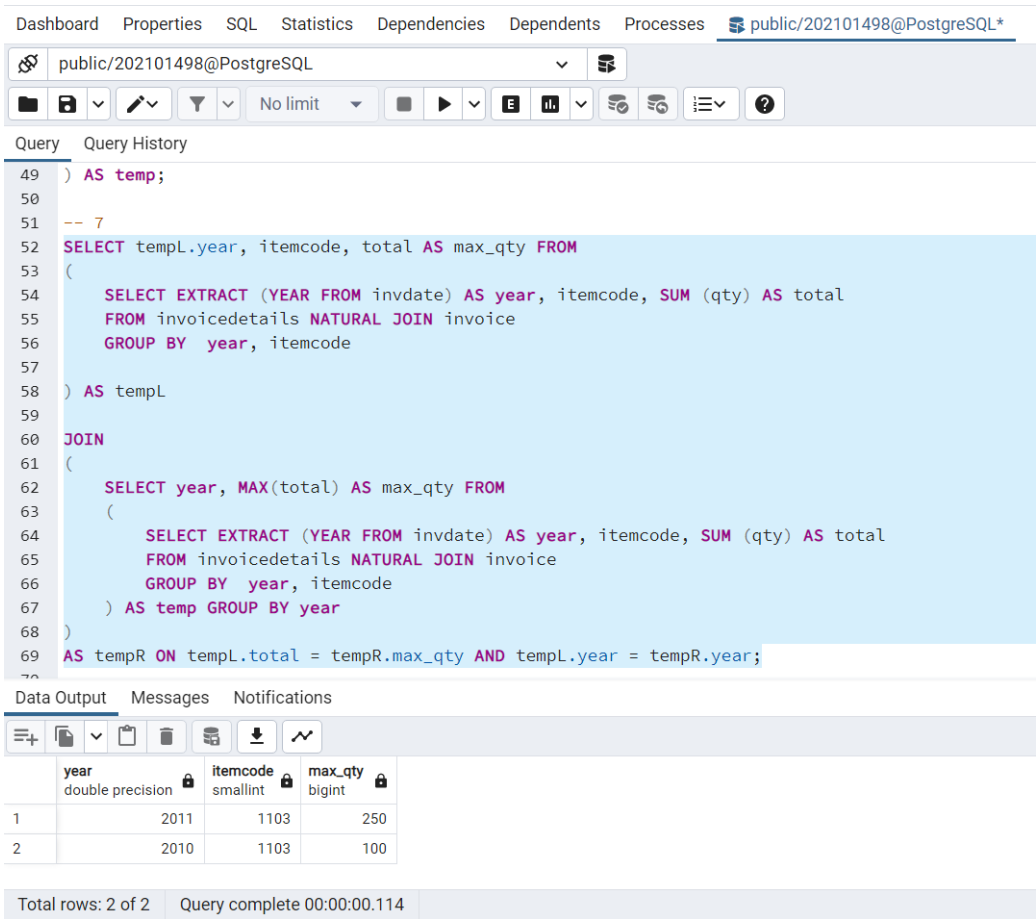
```sql
SELECT custid FROM customer JOIN invoice ON custid = customerid
NATURAL JOIN
(
    SELECT invno, SUM (qty * (rate - averagepurchaseprice))
    AS total_profit FROM invoicedetails
    JOIN items ON code = itemcode
    GROUP BY invno
    ORDER BY total DESC
    LIMIT 1
) AS temp;
```



7. Top selling item (in terms of numbers) for a given year.

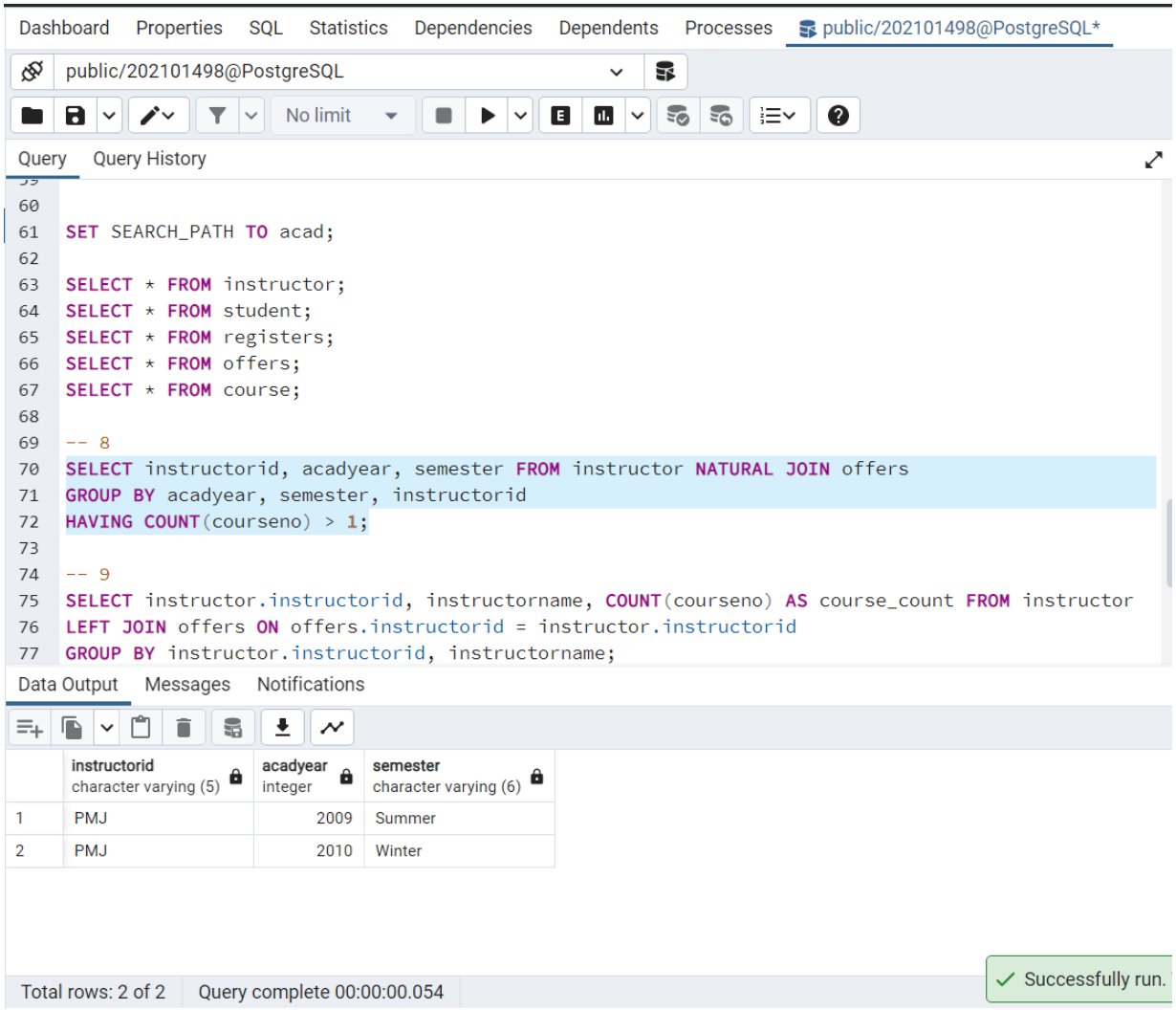
```sql
SELECT tempL.year, itemcode, total AS max_qty FROM
(
        SELECT EXTRACT (YEAR FROM invdate) AS year, itemcode,
        SUM (qty) AS total
        FROM invoicedetails NATURAL JOIN invoice
        GROUP BY  year, itemcode

) AS tempL
JOIN
(
    SELECT year, MAX(total) AS max_qty FROM
    (
        SELECT EXTRACT (YEAR FROM invdate) AS year, itemcode,
        SUM (qty) AS total
        FROM invoicedetails NATURAL JOIN invoice
        GROUP BY  year, itemcode
    ) AS temp GROUP BY year
)
AS tempR ON tempL.total = tempR.max_qty AND
tempL.year = tempR.year;
```



8. Retrieve ID of faculties who took more than one courses in a semester (for all semester in the database)

```sql
SELECT instructorid, acadyear, semester FROM instructor
NATURAL JOIN offers
GROUP BY acadyear, semester, instructorid
HAVING COUNT(courseno) > 1;
```



9. List total count for each instructor. List the faculty name even if course count is zero

```sql
SELECT instructor.instructorid, instructorname, COUNT(courseno)
AS course_count FROM instructor
LEFT JOIN offers ON offers.instructorid = instructor.instructorid
GROUP BY instructor.instructorid, instructorname;
```

Dashboard Properties SQL Statistics Dependencies Dependents Processes public/202101498@PostgreSQL*

public/202101498@PostgreSQL

No limit

Query Query History

```sql
HAVING COUNT(courseno) > 1;

-- 9
SELECT instructor.instructorid, instructorname, COUNT(courseno) AS course_count FROM instructor
LEFT JOIN offers ON offers.instructorid = instructor.instructorid
GROUP BY instructor.instructorid, instructorname;

-- 10
SELECT studentid, name, SUM(credit) AS total_credits FROM student
NATURAL JOIN registers NATURAL JOIN course
WHERE progid = '02' AND batch = 2007 AND semester = 'Autumn' AND acadyear = '2008'
GROUP BY studentid, name;

-- 11
SELECT studentid, name FROM student NATURAL JOIN registers
WHERE semester = 'Autumn' AND acadyear = '2008' AND grade = 'F'
GROUP BY studentid, name
HAVING COUNT(grade) > 2;
```

Data Output Messages Notifications

	instructorid [PK] character varying (5)	instructorname character varying (30)	course_count bigint
1	MHR	Mehul Raval	4
2	TBD	To be Decided	0
3	MK	Manish Khare	0
4	VS	V Sunita	0
5	VKC	Vijay Kumar Chakka	0

Total rows: 11 of 11 Query complete 00:00:00.112

10. Retrieve all students (StudentID, Name, TotalCreditTaken) for B.Tech. (CS) (progid='02') batch 2007 in Autumn'2008

```sql
SELECT studentid, name, SUM(credit) AS total_credits
FROM student  NATURAL JOIN registers NATURAL JOIN course
WHERE progid = '02' AND batch = 2007
AND semester = 'Autumn' AND acadyear = '2008'
GROUP BY studentid, name;
```

Dashboard Properties SQL Statistics Dependencies Dependents Processes public/202101498@PostgreSQL*

public/202101498@PostgreSQL

No limit

Query Query History

```sql
HAVING COUNT(courseno) > 1;

-- 9
SELECT instructor.instructorid, instructorname, COUNT(courseno) AS course_count FROM instructor
LEFT JOIN offers ON offers.instructorid = instructor.instructorid
GROUP BY instructor.instructorid, instructorname;

-- 10
SELECT studentid, name, SUM(credit) AS total_credits FROM student
NATURAL JOIN registers NATURAL JOIN course
WHERE progid = '02' AND batch = 2007 AND semester = 'Autumn' AND acadyear = '2008'
GROUP BY studentid, name;

-- 11
SELECT studentid, name FROM student NATURAL JOIN registers
WHERE semester = 'Autumn' AND acadyear = '2008' AND grade = 'F'
GROUP BY studentid, name
HAVING COUNT(grade) > 2;
```

Data Output Messages Notifications

	studentid [PK] character varying (9)	name character varying (30)	total_credits numeric
1	200702001	Sumit Sharma	4.0
2	200702002	Amit Kumar	4.0
3	200702003	Shobha Kiran	4.0
4	200702004	Raj Gupta	4.0
5	200702005	Amit Tiwari	4.0

Total rows: 5 of 5 Query complete 00:00:00.179

11. Retrieve all students (Id and name) who got more than two F grades in Autumn'2008

```sql
SELECT studentid, name FROM student NATURAL JOIN registers
WHERE semester = 'Autumn' AND acadyear = '2008' AND grade = 'F'
GROUP BY studentid, name
HAVING COUNT(grade) > 2;
```