

DBMS

202412012

JAYESH S CHAUHAN

Query:

Output:

Data Output	Messages	Notifications	
<div> <div>☰</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> <div>SQL</div> </div>			
	name character varying (100)	price numeric (10,2)	stock_quantity integer
1	Hair Dryer	9972.00	51
Total rows: 1 of 1		Query complete 00:00:00.101	

Query:

JAYESH S CHAUHAN

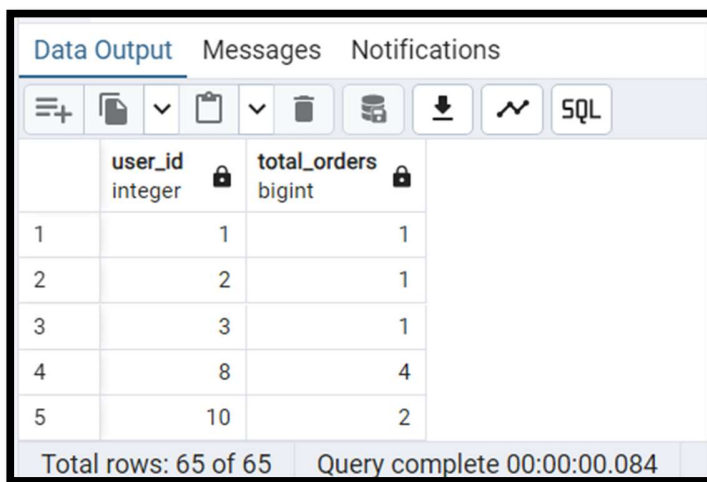
202412012

DBMS

ASSIGNMENT: 2

```
USER_ID,  
COUNT(ORDER_ID) AS TOTAL_ORDERS  
FROM  
"EC_DB".ORDERS  
GROUP BY  
USER_ID  
ORDER BY  
USER_ID;
```

Output:



The screenshot shows a database interface with tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with the following data:

	user_id integer	total_orders bigint
1	1	1
2	2	1
3	3	1
4	8	4
5	10	2

At the bottom of the window, it states 'Total rows: 65 of 65' and 'Query complete 00:00:00.084'.

3. Find the product with the highest number of reviews.

Query:

```
SELECT  
COUNT(REVIEW_ID) AS TOTAL_REVIEWS,  
PRODUCTS.NAME  
FROM  
"EC_DB".REVIEWS  
JOIN "EC_DB".PRODUCTS ON REVIEWS.PRODUCT_ID =  
PRODUCTS.PRODUCT_ID  
GROUP BY  
PRODUCTS.NAME  
ORDER BY  
TOTAL_REVIEWS DESC;
```

Output:

Data Output Messages Notifications			
	total_reviews bigint		name character varying (100)
1	5		Rice
2	5		Shirt
3	5		Doll
4	5		Car Battery
5	4		Headphones
Total rows: 44 of 44		Query complete 00:00:00.124	

4. Get the average rating of each product.**Query:**

```

SELECT
    PRODUCT_ID,
    AVG(RATING) AS AVG_RATING
FROM
    "EC_DB".REVIEWS
GROUP BY
    PRODUCT_ID
ORDER BY
    PRODUCT_ID;

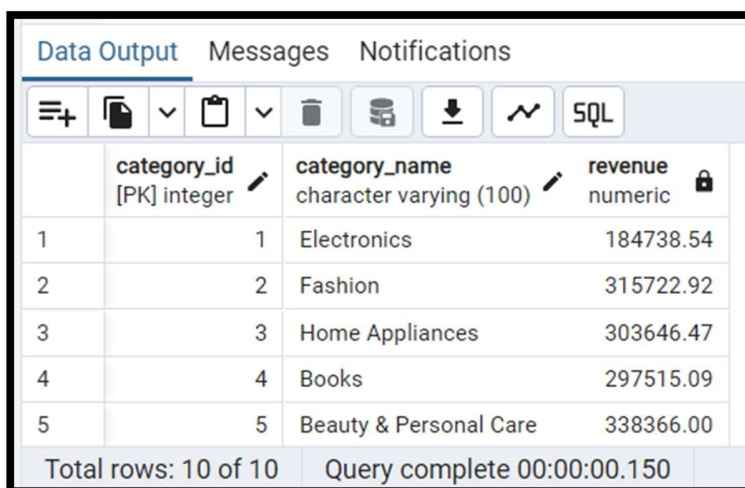
```

Output:

Data Output Messages Notifications			
	product_id integer		avg_rating numeric
1	1		4.0000000000000000
2	2		4.0000000000000000
3	3		2.0000000000000000
4	4		3.0000000000000000
5	6		3.2000000000000000
Total rows: 44 of 44		Query complete 00:00:00.079	

5. Get the total revenue generated by each category.**Query:**

```
SELECT
    CAT.CATEGORY_ID,
    CAT.CATEGORY_NAME,
    SUM(ORD.TOTAL_AMOUNT) REVENUE
FROM
    "EC_DB".ORDERS ORD
    JOIN "EC_DB".ORDER_DETAILS ORD_DET ON ORD.ORDER_ID =
ORD_DET.ORDER_ID
    JOIN "EC_DB".PRODUCTS PRO ON ORD_DET.PRODUCT_ID =
PRO.PRODUCT_ID
    JOIN "EC_DB".CATEGORIES CAT ON PRO.CATEGORY_ID = CAT.CATEGORY_ID
GROUP BY
    CAT.CATEGORY_ID
ORDER BY
    CAT.CATEGORY_ID;
```

Output:

The screenshot shows a database interface with a 'Data Output' tab. It displays a table with 5 rows of data. The columns are 'category_id' (integer, primary key), 'category_name' (character varying (100)), and 'revenue' (numeric). The data is as follows:

	category_id [PK] integer	category_name character varying (100)	revenue numeric
1	1	Electronics	184738.54
2	2	Fashion	315722.92
3	3	Home Appliances	303646.47
4	4	Books	297515.09
5	5	Beauty & Personal Care	338366.00

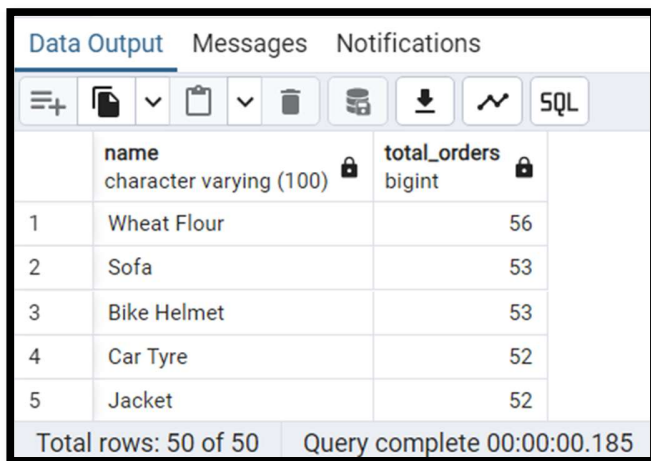
At the bottom, it shows 'Total rows: 10 of 10' and 'Query complete 00:00:00.150'.

6. Find the most popular product (most ordered).**Query:**

```
SELECT
```

DBMS**ASSIGNMENT: 2**

```
PRO.NAME,  
SUM(ORD_DET.QUANTITY) AS TOTAL_ORDERS  
FROM  
"EC_DB".PRODUCTS PRO  
JOIN "EC_DB".ORDER_DETAILS ORD_DET ON ORD_DET.PRODUCT_ID =  
PRO.PRODUCT_ID  
GROUP BY  
PRO.NAME  
ORDER BY  
TOTAL_ORDERS DESC;
```

Output:

The screenshot shows a database interface with tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with two columns: 'name' (character varying (100)) and 'total_orders' (bigint). The table contains 5 rows of data. Below the table, it indicates 'Total rows: 50 of 50' and 'Query complete 00:00:00.185'.

	name character varying (100)	total_orders bigint
1	Wheat Flour	56
2	Sofa	53
3	Bike Helmet	53
4	Car Tyre	52
5	Jacket	52

Total rows: 50 of 50 Query complete 00:00:00.185

7. List users who have spent more than \$1000 in total.**Query:**

```
SELECT  
U.USER_ID,  
U.USERNAME,  
SUM(O.TOTAL_AMOUNT) SPENTED_AMOUNT  
FROM  
"EC_DB".USERS U  
JOIN "EC_DB".ORDERS O ON U.USER_ID = O.USER_ID  
WHERE  
O.TOTAL_AMOUNT > 1000
```

DBMS

ASSIGNMENT: 2


GROUP BY

U.USER_ID

ORDER BY

U.USER_ID;

Output:

Data Output Messages Notifications			
			
	user_id [PK] integer	username character varying (50)	spented_amount numeric
1	1	mamooty17	19371.83
2	2	stuvantara	18724.36
3	3	hrishita94	16212.64
4	8	rgola	53060.24
5	10	siyengar	9201.82
Total rows: 63 of 63		Query complete 00:00:00.092	

8. Get the details of the largest order (by total amount).

Query:

SELECT

*

FROM

"EC_DB".ORDER_DETAILS ORD_DET

JOIN "EC_DB".ORDERS ORD ON ORD_DET.ORDER_ID = ORD.ORDER_ID

WHERE

ORD.TOTAL_AMOUNT = (

SELECT

MAX(TOTAL_AMOUNT)

FROM

"EC_DB".ORDERS

);

Output:

Data Output Messages Notifications									
	order_detail_id integer	order_id integer	product_id integer	quantity integer	price numeric (10,2)	order_id integer	user_id integer	order_date timestamp without time zone	
1	95	4	35	2	1162.28	4	74	2024-03-24 00:00:00	
2	147	4	35	2	5721.03	4	74	2024-03-24 00:00:00	
3	241	4	40	1	2917.16	4	74	2024-03-24 00:00:00	
Total rows: 3 of 3 Query complete 00:00:00.076									

9. List all products that have never been ordered.

Query:

Output:

10. Find the most active user (by the number of reviews).

Query:

```
SELECT
    COUNT(REV.REVIEW_ID) AS REVIEW_COUNT,
    US.USERNAME
FROM
    "EC_DB".REVIEWS REV
    JOIN "EC_DB".USERS US ON REV.USER_ID = US.USER_ID
GROUP BY
    US.USERNAME
ORDER BY
    REVIEW_COUNT DESC
LIMIT
    1
```

Output:

Data Output			Messages	Notifications
	review_count bigint	username character varying (50)		
1	4	siyengar		
Total rows: 1 of 1			Query complete 00:00:00.074	

11. Get the average order amount per user.

Query:

```

SELECT
    US.USERNAME,
    AVG(ORD.TOTAL_AMOUNT) AVG_ORDERED_AMOUNT
FROM
    "EC_DB".ORDERS ORD
    JOIN "EC_DB".USERS US ON ORD.USER_ID = US.USER_ID
GROUP BY
    US.USERNAME;

```

Output:

Data Output			Messages	Notifications
	username character varying (50)	avg_total_amount numeric		
1	raushlok	13374.1400000000000000		
2	ravalyuvraj	3205.4950000000000000		
3	vivaanandra	14764.5900000000000000		
4	rbhardwaj	15494.9600000000000000		
5	umangdin	1383.1500000000000000		
Total rows: 65 of 65			Query complete 00:00:00.066	

12. List all users who have not placed any orders.

Query:

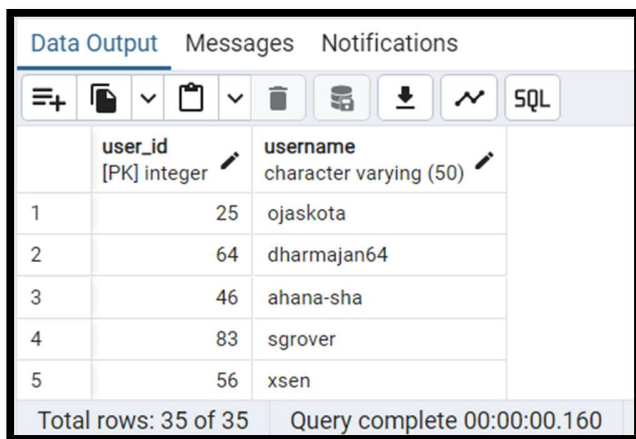
```

SELECT

```


DBMS**ASSIGNMENT: 2**

```
US.USER_ID,  
US.USERNAME  
FROM  
"EC_DB".ORDERS ORD  
RIGHT OUTER JOIN "EC_DB".USERS US ON US.USER_ID = ORD.USER_ID  
WHERE  
ORD.ORDER_ID IS NULL;
```

Output:

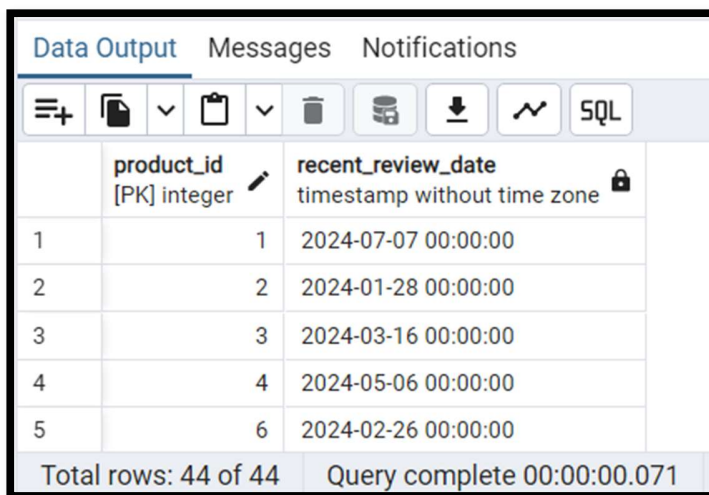
The screenshot shows a database interface with tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with two columns: 'user_id' (integer, primary key) and 'username' (character varying (50)). The table contains 5 rows of data. Below the table, it indicates 'Total rows: 35 of 35' and 'Query complete 00:00:00.160'.

	user_id [PK] integer	username character varying (50)
1	25	ojaskota
2	64	dharmajan64
3	46	ahana-sha
4	83	sgrover
5	56	xsen

13. Find the most recent review for each product.**Query:**

```
SELECT  
PD.PRODUCT_ID,  
MIN(REV.REVIEW_DATE) AS RECENT_REVIEW_DATE  
FROM  
"EC_DB".REVIEWS REV  
JOIN "EC_DB".PRODUCTS PD ON REV.PRODUCT_ID = PD.PRODUCT_ID  
GROUP BY  
PD.PRODUCT_ID  
ORDER BY  
PD.PRODUCT_ID;
```

Output:



The screenshot shows a database management interface with tabs for 'Data Output', 'Messages', and 'Notifications'. Below the tabs is a toolbar with icons for various database operations. The main area displays a table with two columns: 'product_id' (integer, primary key) and 'recent_review_date' (timestamp without time zone). The table contains five rows of data. At the bottom, a status bar indicates 'Total rows: 44 of 44' and 'Query complete 00:00:00.071'.

	product_id [PK] integer	recent_review_date timestamp without time zone
1	1	2024-07-07 00:00:00
2	2	2024-01-28 00:00:00
3	3	2024-03-16 00:00:00
4	4	2024-05-06 00:00:00
5	6	2024-02-26 00:00:00

Total rows: 44 of 44 Query complete 00:00:00.071

14. Get the list of users who have reviewed all products they have purchased.

Query:

SELECT

REV.REVIEW_ID,
REV.USER_ID,
REV.PRODUCT_ID,
ORD.ORDER_ID,
ORD.USER_ID

FROM

"EC_DB".REVIEWS REV

RIGHT OUTER JOIN "EC_DB".ORDERS ORD ON REV.USER_ID = ORD.USER_ID

WHERE

REVIEW_ID IS NOT NULL

Output:

Data Output Messages Notifications						
	review_id integer	user_id integer	product_id integer	order_id integer	user_id integer	
1	1	99	43	21	99	
2	2	73	19	86	73	
3	2	73	19	58	73	
4	5	78	7	36	78	
5	7	79	26	28	79	
Total rows: 82 of 82 Query complete 00:00:00.061						

15. Create a view that shows the total amount spent by each user.

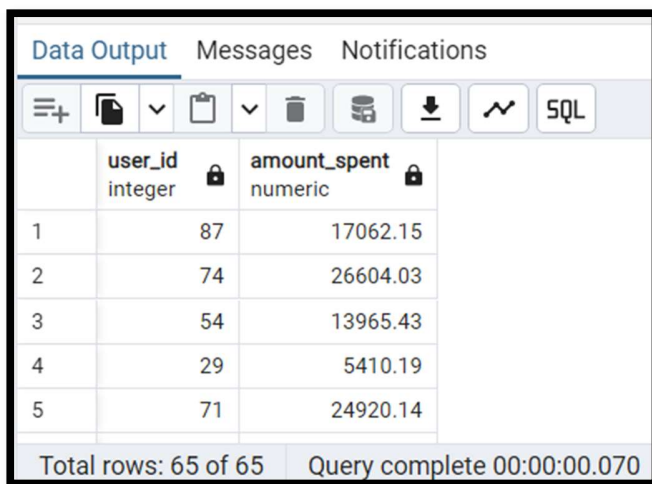
Query:

```
CREATE VIEW TOTAL_AMOUNT_SPENT_BY_EACH_USER AS
SELECT
    USER_ID,
    SUM(TOTAL_AMOUNT) AS AMOUNT_SPENT
FROM
    "EC_DB".ORDERS
GROUP BY
    USER_ID
```

```
SELECT * FROM TOTAL_AMOUNT_SPENT_BY_EACH_USER;
```

Output:

Data Output Messages Notifications
CREATE VIEW
Query returned successfully in 99 msec.



The screenshot shows a database query result window with tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with two columns: 'user_id' (integer) and 'amount_spent' (numeric). The table contains five rows of data. Below the table, it indicates 'Total rows: 65 of 65' and 'Query complete 00:00:00.070'.

	user_id integer	amount_spent numeric
1	87	17062.15
2	74	26604.03
3	54	13965.43
4	29	5410.19
5	71	24920.14












Total rows: 65 of 65 Query complete 00:00:00.070

16. Get the average price of products in each category that have been reviewed.

Query:

```
SELECT
    CAT.CATEGORY_ID,
    AVG(PRO.PRICE) AS AVG_PRICE
FROM
    "EC_DB".PRODUCTS PRO
    JOIN "EC_DB".CATEGORIES CAT ON CAT.CATEGORY_ID = PRO.CATEGORY_ID
WHERE
    PRO.PRODUCT_ID IN (
        SELECT DISTINCT
            REVIEWS.PRODUCT_ID
        FROM
            "EC_DB".REVIEWS
    )
GROUP BY
    CAT.CATEGORY_ID
ORDER BY
    CAT.CATEGORY_ID
```

Output:

Data Output Messages Notifications			
         SQL			
	category_id [PK] integer 	avg_price numeric 	
1	1	4843.8575000000000000	
2	2	2567.1840000000000000	
3	3	4258.5500000000000000	
4	4	2986.6860000000000000	
5	5	7851.6300000000000000	
Total rows: 10 of 10		Query complete 00:00:00.091	

17. List all users who have placed more than 2 orders.

Query:

```
SELECT
    ORD.USER_ID,
    COUNT(ORD.ORDER_ID)
FROM
    "EC_DB".ORDERS ORD
GROUP BY
    ORD.USER_ID
HAVING
    COUNT(ORD.ORDER_ID) > 2;
```

Output:

Data Output					Messages	Notifications
	user_id		count			
	integer		bigint			
1	59		3			
2	75		3			
3	88		3			
4	40		3			
5	8		4			
Total rows: 5 of 5					Query complete 00:00:00.122	

18. Find users who have made purchases from all categories.

Query:

```
SELECT
    US.USER_ID,
    US.USERNAME
FROM
    "EC_DB".USERS US
    JOIN "EC_DB".ORDERS ORD ON US.USER_ID = ORD.USER_ID
    JOIN "EC_DB".ORDER_DETAILS ORD_DET ON ORD_DET.ORDER_ID =
ORD.ORDER_ID
    JOIN "EC_DB".PRODUCTS PD ON ORD_DET.PRODUCT_ID = PD.PRODUCT_ID
    LEFT JOIN "EC_DB".CATEGORIES CAT ON PD.CATEGORY_ID =
CAT.CATEGORY_ID
WHERE
    CAT.CATEGORY_ID IS NOT NULL
GROUP BY
    US.USER_ID,
    US.USERNAME
ORDER BY
    US.USER_ID;
```

Output:

Data Output

Messages

Notifications

≡

+

▼

▼

SQL

	<div>name</div> <div>character varying (100)</div> <div></div>	<div>price</div> <div>numeric (10,2)</div> <div></div>	<div>stock_quantity</div> <div>integer</div> <div></div>
1	Hair Dryer	9972.00	51

Total rows: 1 of 1

Query complete 00:00:00.071

19. List all users who have placed orders in the last 30 days but haven't made a purchase in the last 7 days.

Query:

SELECT

*

FROM

"EC_DB".ORDERS

WHERE

ORDER_DATE > CURRENT_DATE -30

AND ORDER_DATE < CURRENT_DATE -7

Output:

Data Output

Messages

Notifications

SQL

	order_id [PK] integer	user_id integer	order_date timestamp without time zone	shipping_address text	total_amount numeric (10,2)	status character varying (50)
1	9	89	2024-07-26 00:00:00	97/92, Chhabra Zila, Nagaon-085732	8472.91	Pending
2	29	29	2024-07-23 00:00:00	H.No. 392, Kunda Marg, Bareilly 871521	4842.46	Cancelled
3	38	54	2024-07-23 00:00:00	83/609, Barad, Sasaram 479972	7250.11	Delivered
4	75	75	2024-07-29 00:00:00	02, Johal Street, Tirunelveli-120696	17654.42	Shipped
5	82	17	2024-08-02 00:00:00	05/57, Lala Path, Miryalaguda 003294	13591.18	Pending
Total rows: 7 of 7		Query complete 00:00:00.100				

20. Identify the products with a stock quantity below the average stock level.

Query:

SELECT

*

FROM

DBMS**ASSIGNMENT: 2**

```
"EC_DB".PRODUCTS
WHERE
  STOCK_QUANTITY <= (
    SELECT
      AVG(STOCK_QUANTITY)
    FROM
      "EC_DB".PRODUCTS
  )
```

Output:

Data Output Messages Notifications						
SQL						
	product_id [PK] integer	name character varying (100)	description text	price numeric (10,2)	stock_quantity integer	category_id integer
1	1	Mobile Phone	Portable communication device with internet access and app...	7498.65	28	1
2	4	Headphones	Audio device for listening to music or calls privately.	6992.10	18	1
3	6	Shirt	Upper body garment with sleeves and a collar.	5909.62	32	2
4	8	Shoes	Footwear for protection and style.	2334.15	30	2
5	13	Refrigerator	Appliance for storing food at low temperatures.	9781.32	21	3
Total rows: 24 of 24 Query complete 00:00:00.126						

21. Create a view 'product_avg_rating' to show the average rating of each product**Query:**

```
CREATE VIEW PRODUCT_AVG_RATING AS
SELECT
  ROUND(AVG(REV.RATING), 2),
  PD.NAME
FROM
  "EC_DB".REVIEWS REV
  JOIN "EC_DB".PRODUCTS PD ON REV.PRODUCT_ID = PD.PRODUCT_ID
GROUP BY
  PD.NAME
```

```
SELECT * FROM PRODUCT_AVG_RATING;
```

Output:

DBMS

Data Output	Messages	Notifications
CREATE VIEW		
Query returned successfully in 57 msec.		

ASSIGNMENT: 2

Data Output	Messages	Notifications
<div>SQL</div>		
	round numeric	name character varying (100)
1	3.67	Dumbbells
2	4.00	Novel
3	1.00	Wheat Flour
4	3.33	Dress
5	2.50	Toy Car
Total rows: 44 of 44		Query complete 00:00:00.070

22. List products with an average rating above 4.5 using the 'product_avg_rating' view.

Query:

SELECT

*

FROM

PRODUCT_AVG_RATING

WHERE

ROUND > 4.5

Output:

Data Output	Messages	Notifications
<div>SQL</div>		
	round numeric	name character varying (100)
1	4.67	Perfume
2	5.00	Yoga Mat
3	4.67	Textbook
4	5.00	Bike Helmet
5	5.00	Exercise Bike
Total rows: 7 of 7		Query complete 00:00:00.114

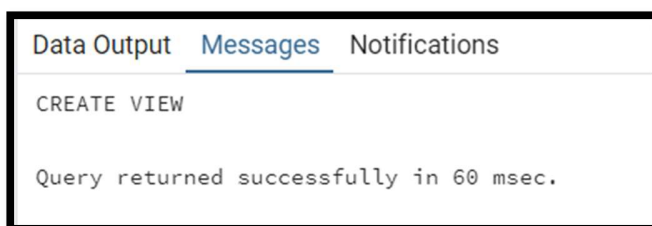
23. Create a view 'product_review_count' to show the number of reviews per product.

Query:

```
CREATE VIEW PRODUCT_REVIEW_COUNT AS
SELECT
    COUNT(REV.REVIEW_ID) AS TOTAL_REVIEWS,
    PRODUCT_ID
FROM
    "EC_DB".REVIEWS REV
GROUP BY
    PRODUCT_ID
```

```
SELECT * FROM PRODUCT_REVIEW_COUNT;
```

Output:



The screenshot shows the 'Data Output' tab in SQL Developer. It displays the result of the query 'SELECT * FROM PRODUCT_REVIEW_COUNT;'. The table has two columns: 'total_reviews' (bigint) and 'product_id' (integer). There are 5 rows of data.

	total_reviews bigint	product_id integer
1	1	42
2	3	29
3	4	4
4	2	34
5	2	41

Total rows: 44 of 44 Query complete 00:00:00.086

24. List products with more than 10 reviews using the 'product_review_count' view.

Query:

```
SELECT
    *
FROM
    PRODUCT_REVIEW_COUNT
WHERE
    TOTAL_REVIEWS > 10;
```

Output:

Data Output	Messages	Notifications
<div> <div>≡</div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗑️</div> <div>📥</div> <div>📥</div> <div>📈</div> <div>SQL</div> </div>	<div>total_reviews</div> <div>bigint</div> <div>🔒</div>	<div>product_id</div> <div>integer</div> <div>🔒</div>
<div>Total rows: 0 of 0</div> <div>Query complete 00:00:00.129</div>		

25. Create a view 'recent_orders' to show all orders placed in the last 30 days.

Query:

```
CREATE VIEW RECENT_ORDERS AS
```

```
SELECT
```

```
    *
```

```
FROM
```

```
    "EC_DB".ORDERS
```

```
WHERE
```

```
    ORDER_DATE > CURRENT_DATE -30
```

```
SELECT * FROM RECENT_ORDERS;
```

Output:

Data Output	Messages	Notifications
<div>CREATE VIEW</div>		
<div>Query returned successfully in 58 msec.</div>		

Data Output

Messages

Notifications

≡

+

▼

▼

SQL

	order_id integer	user_id integer	order_date timestamp without time zone	shipping_address text	total_amount numeric (10,2)	status character varying (50)
1	9	89	2024-07-26 00:00:00	97/92, Chhabra Zila, Nagaon-085732	8472.91	Pending
2	29	29	2024-07-23 00:00:00	H.No. 392, Kunda Marg, Bareilly 871521	4842.46	Cancelled
3	38	54	2024-07-23 00:00:00	83/609, Barad, Sasaram 479972	7250.11	Delivered
4	75	75	2024-07-29 00:00:00	02, Johal Street, Tirunelveli-120696	17654.42	Shipped
5	82	17	2024-08-02 00:00:00	05/57, Lala Path, Miryalaguda 003294	13591.18	Pending

Total rows: 7 of 7

Query complete 00:00:00.087

26. List the most expensive orders placed in the last 30 days using the 'recent_orders' view.

Query:

```
SELECT
    *
FROM
    RECENT_ORDERS
WHERE
    TOTAL_AMOUNT = (
        SELECT
            MAX(TOTAL_AMOUNT)
        FROM
            RECENT_ORDERS
    )
```

Output:

Data Output Messages Notifications							
	order_id integer	user_id integer	order_date timestamp without time zone	shipping_address text	total_amount numeric (10,2)	status character varying (50)	
1	75	75	2024-07-29 00:00:00	02, Johal Street, Tirunelveli-120696	17654.42	Shipped	
Total rows: 1 of 1 Query complete 00:00:00.133							

27. Create a view 'product_sales_quantity' to show the total quantity sold for each product.

Query:

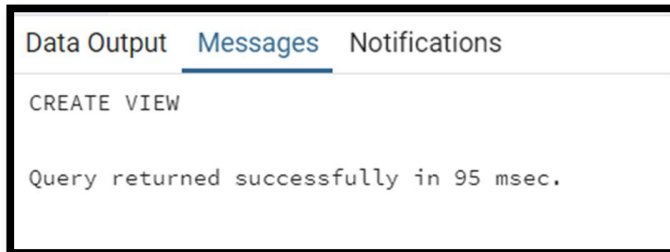
```
CREATE VIEW PRODUCT_SALES_QUANTITY AS
SELECT
    PRODUCT_ID,
    QUANTITY
FROM
    "EC_DB".ORDER_DETAILS
```

DBMS

ASSIGNMENT: 2

SELECT * FROM PRODUCT_SALES_QUANTITY;

Output:



The screenshot shows a database interface with three tabs: 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is selected and highlighted with a blue underline. Below the tabs is a toolbar with various icons. The main area displays a table with two columns: 'product_id' (integer) and 'quantity' (integer). The table contains five rows of data. At the bottom, a status bar indicates 'Total rows: 300 of 300' and 'Query complete 00:00:00.101'.

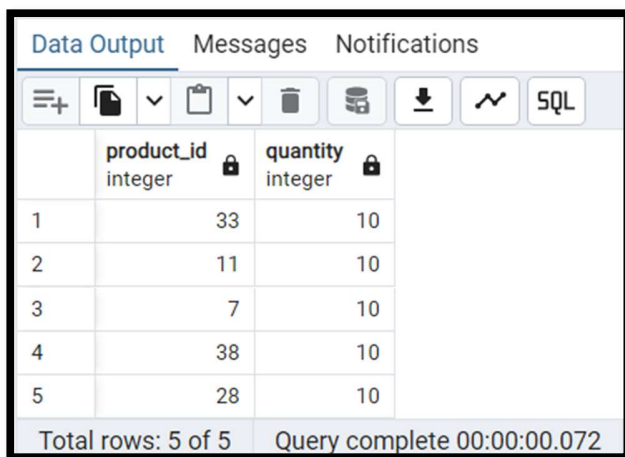
	product_id integer	quantity integer
1	7	10
2	17	1
3	15	2
4	33	8
5	48	6

28. List the top 5 best-selling products using the 'product_sales_quantity' view.

Query:

```
SELECT
    *
FROM
    PRODUCT_SALES_QUANTITY
ORDER BY
    QUANTITY DESC
LIMIT
    5
```

Output:



	product_id integer	quantity integer	
1	33	10	
2	11	10	
3	7	10	
4	38	10	
5	28	10	

Total rows: 5 of 5 Query complete 00:00:00.072

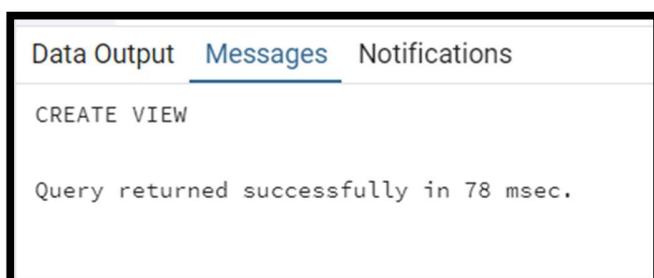
29. Create a view 'product_sales_quantity' to show the total number of orders each user has placed.

Query:

```
CREATE VIEW PRODUCT_SALES_QUANTITY AS
SELECT
    USER_ID,
    COUNT(ORDER_ID) AS TOTAL_ORDERS
FROM
    "EC_DB".ORDERS
GROUP BY
    USER_ID
ORDER BY
    TOTAL_ORDERS DESC
```

```
SELECT * FROM PRODUCT_SALES_QUANTITY
```

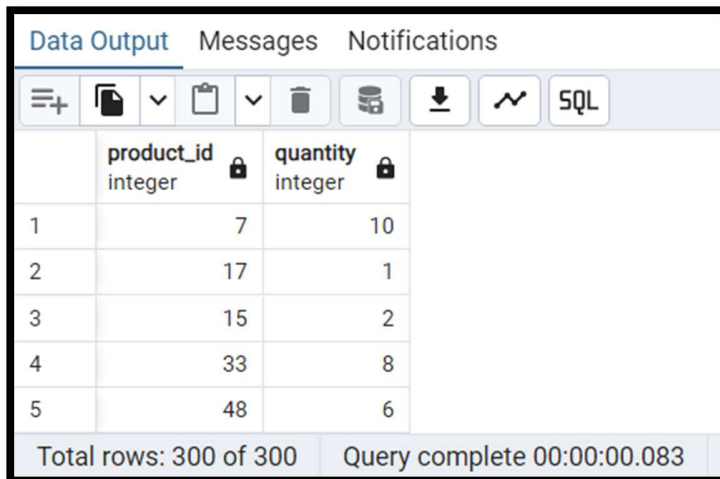
Output:



Data Output	Messages	Notifications
CREATE VIEW		
Query returned successfully in 78 msec.		

DBMS

ASSIGNMENT: 2



The screenshot shows a database interface with tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with two columns: 'product_id' (integer) and 'quantity' (integer). The table contains five rows of data. At the bottom, it indicates 'Total rows: 300 of 300' and 'Query complete 00:00:00.083'.

	product_id integer	quantity integer
1	7	10
2	17	1
3	15	2
4	33	8
5	48	6

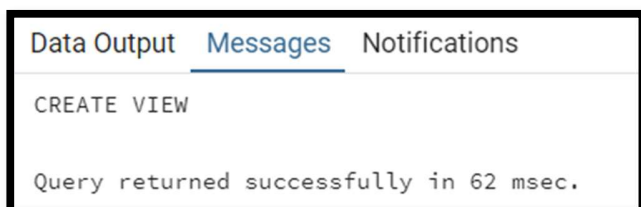
30. List users who have placed more than 5 orders using the 'user_order_count' view.

Query:

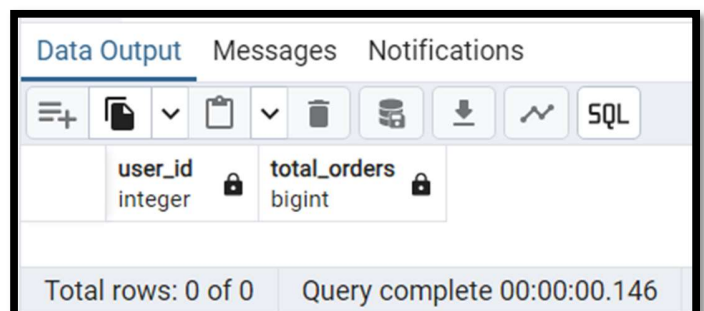
```
CREATE VIEW USER_ORDER_COUNT AS
SELECT
    USER_ID,
    COUNT(ORDER_ID) AS TOTAL_ORDERS
FROM
    "EC_DB".ORDERS
GROUP BY
    USER_ID
ORDER BY
    TOTAL_ORDERS DESC;
```

```
SELECT * FROM USER_ORDER_COUNT WHERE TOTAL_ORDERS > 5;
```

Output:



The screenshot shows a database interface with tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, displaying the text 'CREATE VIEW' and 'Query returned successfully in 62 msec.'



The screenshot shows a database interface with tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table with two columns: 'user_id' (integer) and 'total_orders' (bigint). The table is empty. At the bottom, it indicates 'Total rows: 0 of 0' and 'Query complete 00:00:00.146'.

user_id integer	total_orders bigint
--------------------	------------------------

