

**DBMS - LAB 5**  
**Procedures Functions and views**  
**Art Gallery Management (AGM)**

**NAME : Siddharth Tewari**

**SRN : PES2UG21CS525**

**SEC: I**

**Lab 5 Exercises**

**Task 1:**

**Schema 1:**

1. 

```
CREATE VIEW v2 AS
SELECT books.book_id, title, author_name
FROM books
JOIN authors ON books.author_id = authors.author_id
LEFT JOIN borrows ON books.book_id = borrows.book_id
WHERE borrows.book_id IS NULL OR borrows.return_date <
CURDATE();
```

What does the above view do? what is the output of view when we run select \* from v2;

**ANSWER:**

The view "v2" is designed to retrieve a list of books that are either not borrowed by any user or have been borrowed but are overdue.

When you run a query like `SELECT * FROM v2`, it will return a result set with the following columns:

- `Book_id` : The unique identifier for each book.
- `Title`: The title of the book.
- `Author_name`: The name of the author.

When you run the query `SELECT * FROM v2`, it will display the output that contains the `book_id`, `title`, and `author_name` of the books that match the above criteria. This view can be used to quickly identify books that are either available for borrowing or overdue in the library.

```
2. CREATE VIEW read_only_books AS
SELECT
b.book_id, b.title, a.author_name
FROM books b
JOIN authors a ON b.author_id = a.author_id;

INSERT INTO read_only_books (book_id, title, author_name) VALUES (3, 'New Book', 'John Doe');
```

Will the insert query work? If Yes what is the effect. If NO why ?

### ANSWER:

No , Query will not work.

The insert query you provided attempts to insert a new record into the `read_only_books` view, which is defined as a result of a `SELECT` statement and does not directly correspond to an underlying table. Therefore, this insert query will not work, and it will result in an error.

The reason is that the `read_only_books` view does not represent a physical table with a one-to-one relationship to the underlying data in the `books` and `authors` tables.

To insert data, you would need to insert it directly into the underlying tables, such as `books` and `authors`, not the view.

**Schema 2:**

```
3. CREATE FUNCTION fun(p_category_id INT) RETURNS DECIMAL deterministic
    DECLARE total_sales DECIMAL;

    SELECT SUM(p.price * o.quantity) INTO total_sales
    FROM products p
    JOIN orders o ON p.product_id = o.product_id
    WHERE p.category_id = p_category_id;

    IF total_sales IS NULL THEN
    SET total_sales = 0;
    END IF;

    RETURN total_sales; END;
```

Will the function fun be created without throwing an error? If yes, what does it return for p\_category\_id= 1? If no, what is causing the error?

**ANSWER:**

Based on the supplied "p\_category\_id," the SQL function "fun" is intended to determine the total sales for a specific product category. With the selected category as a filter, a SQL query is used to combine the "products" and "orders" databases. The sum of the product price times the quantity in each order is used to determine the total sales. It returns zero if the category has no sales. If the database schema is set up correctly, the function should run without issues and If there are no sales for category 1, it will return 0, as specified in the function.

```
4. CREATE PROCEDURE fun( IN p_product_id INT, IN p_new_price
```

```
DECIMAL(10, 2)) BEGIN
    DECLARE product_count INT;
    SELECT COUNT(*) INTO product_count
    FROM products
    WHERE product_id = p_product_id;

    IF product_count > 0 THEN
        UPDATE products
        SET price = p_new_price
        WHERE product_id = p_product_id;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Product not found', MYSQL_ERRNO = 1001;
    END IF;
END
Call fun(3,200)
```

Will the procedure fun be created without throwing an error? If yes, what does it do? If not, what is causing the error?

**ANSWER:**

The purpose of the SQL procedure "fun" is to change the price of a product with a given product\_id. It requires two input parameters, looks up whether a product with the specified product\_id already exists, then adjusts the product's pricing accordingly if it does. A custom error with message "Product not found" and the MySQL error code 1001 is raised if the product cannot be located. If there are no syntax mistakes and the structure of the database's "products" table matches what is intended, the method will build without errors. It tries to update the product with "product\_id" 3 to a price of 200 when called with CALL fun(3, 200), or it raises an error if the product doesn't exist.

```
5. CREATE VIEW vl AS
    SELECT c.category_name, AVG(p.price) AS average_price
    FROM categories c
```

```
JOIN products p ON c.category_id = p.category_id  
GROUP BY c.category_name;
```

```
INSERT INTO vl (category_name, average_price) VALUES ('New Category', 50.0);
```

Will the insert query work? If Yes what is the effect. If NO why ?

### ANSWER:

Your provided INSERT query won't function as intended. Your attempt to insert a new record into a view is the cause of this error because views are generally not intended to be directly updated by INSERT, UPDATE, or DELETE actions.

Views do not directly correspond to physical tables; rather, they are virtual tables produced by SELECT queries. They are used for data retrieval and querying, but they cannot be written to. Because the database system is unable to decide how to carry out the insertion into the underlying base tables, you will generally run into errors while attempting to insert into a view. When you wish to add data to a table, you should do the INSERT operation on the actual table. In this case, the table that is associated to the "categories" and "products," not the view.

### Task 2:

6) Imagine you are a curator at the gallery, and you want to keep a close eye on the status of the artworks displayed in your gallery. You've created a view called `ArtworkDetails` which provides comprehensive information about each artwork, including its ID, description, artwork name, artist's ID, artist's location, gallery ID, gallery name, gallery location, and an availability status indicating whether the artwork has been ordered or not. How would you use this view to identify which artworks in your gallery have not yet been ordered, helping you decide which ones to promote more actively to potential customers?

## ANSWER:

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- Indexes
- Foreign Keys
- Triggers
- 118\_art\_backup
- 118\_art\_order
- 118\_artist
- 118\_comments
- 118\_customer
- 118\_exhibited
- 118\_exhibition
- 118\_gallery
- 118\_payment
- 118\_purchase
- 118\_specialization
- Views
  - artworkdetails
- Stored Procedures
- Functions
- art\_gallery
- art\_gallery\_management\_system
- artgallery\_management\_system\_1:
- company
- db

Administration Schemas

Information:

Schema: db

SQL File 4\* artworkdetails

```

1 /*6(1)*/
2 CREATE VIEW ArtworkDetails AS
3 SELECT
4     a.art_id,
5     a.art_description,
6     a.artist_id,
7     a.gallery_id,
8     a.availability,
9     CONCAT(118_art.f_name,',',118_art.l_name) AS artist_name,
10    118_art.location AS artist_location,
11    g.g_name AS gallery_name,
12    g.g_location AS gallery_location
13 FROM 118_art a
14 LEFT JOIN 118_artist 118_art ON a.artist_id = 118_art.artist_id
15 LEFT JOIN 118_gallery g ON a.gallery_id = g.g_id;

```

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- Indexes
- Foreign Keys
- Triggers
- 118\_art\_backup
- 118\_art\_order
- 118\_artist
- 118\_comments
- 118\_customer
- 118\_exhibited
- 118\_exhibition
- 118\_gallery
- 118\_payment
- 118\_purchase
- 118\_specialization
- Views
  - artworkdetails
- Stored Procedures
- Functions
- art\_gallery
- art\_gallery\_management\_system
- artgallery\_management\_system\_1:
- company
- db

Administration Schemas

Information:

Schema: db

SQL File 4\* artworkdetails

```

1 SELECT * FROM 118_art_gallery_db.artworkdetails;

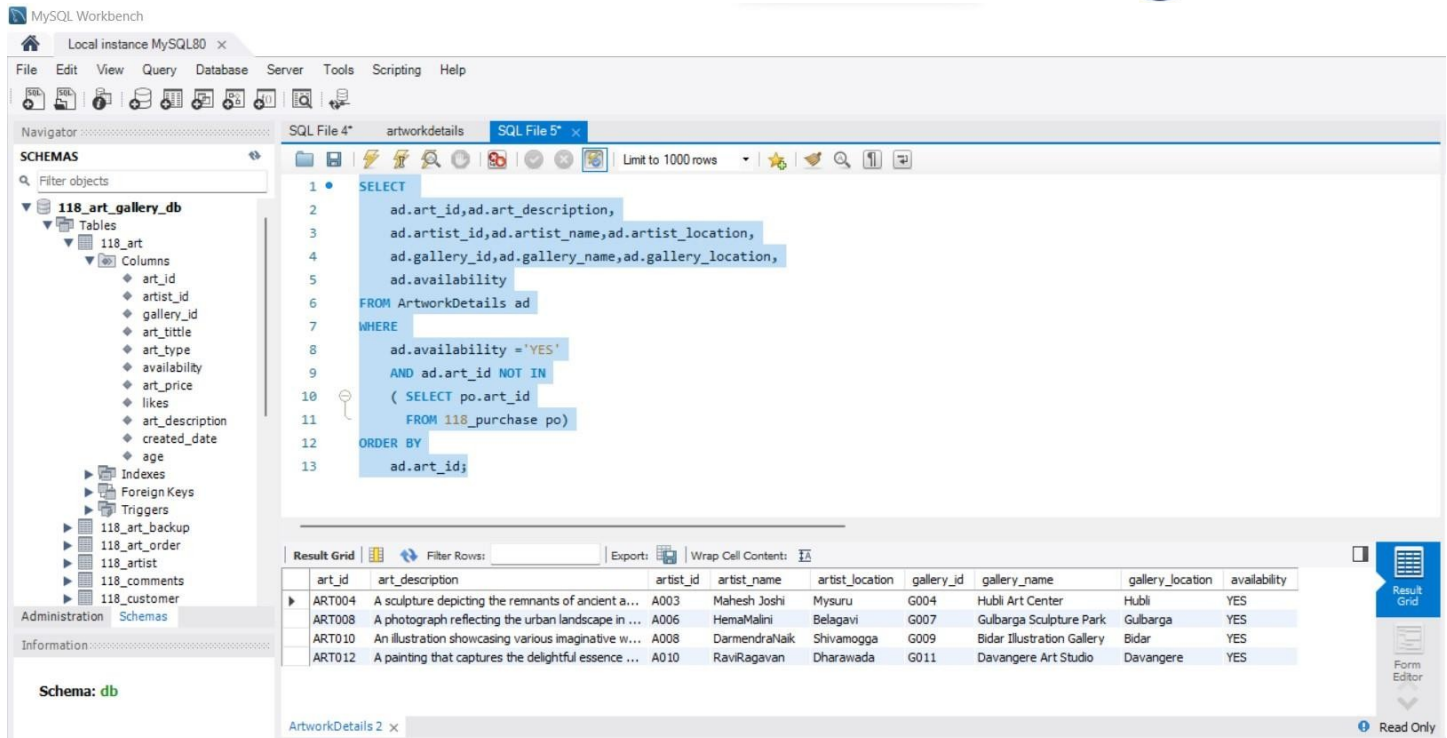
```

Result Grid

art_id	art_description	artist_id	gallery_id	availability	artist_name	artist_location	gallery_name	gallery_location
ART001	A serene painting capturing the beauty of the s...	A001	G002	YES	GuruDev	Bangaluru	Bangaluru Chitra Kala Parishat	Bengaluru
ART002	A stunning photograph showcasing the city's sk...	A002	G003	YES	KaranDesai	Magaluru	Mysuru Art Haven	Mysuru
ART003	A sculpture that represents the harmony betwe...	A002	G003	YES	KaranDesai	Magaluru	Mysuru Art Haven	Mysuru
ART004	A sculpture depicting the remnants of ancient a...	A003	G004	YES	Mahesh Joshi	Mysuru	Hubli Art Center	Hubli
ART005	A beautiful painting capturing the breathtaking ...	A004	G005	YES	ManikantanMurugan	Bengaluru	Belgaum Art Gallery	Belgaum
ART006	A painting that showcases the elegance of vari...	A004	G005	YES	ManikantanMurugan	Bengaluru	Belgaum Art Gallery	Belgaum
ART007	A photograph capturing the serene atmosphere...	A005	G006	YES	JayaLalitha	Mysuru	Mangaluru Art Studio	Mangaluru
ART008	A photograph reflecting the urban landscape in ...	A006	G007	YES	HemaMalini	Belagavi	Gulbarga Sculpture Park	Gulbarga
ART009	A digital artwork that portrays a dreamlike fant...	A007	G008	YES	StevGarcia	Bengaluru	Udupi Digital Art Showcase	Udupi
ART010	An illustration showcasing various imaginative w...	A008	G009	YES	DamendraNaik	Shivamogga	Bidar Illustration Gallery	Bidar
ART011	Printed landscapes that depict the beauty of dif...	A009	G010	YES	Lataparamesh	Hubballi	Dharwad Printmaking Workshop	Dharwad
ART012	A painting that captures the delightful essence ...	A010	G011	YES	RaviRagavan	Dharwad	Davangere Art Studio	Davangere
ART013	A photograph capturing the blissful moments in ...	A011	G002	YES	MahadevMani	Davangere	Bangaluru Chitra Kala Parishat	Bengaluru

artworkdetails 1 x

Read Only



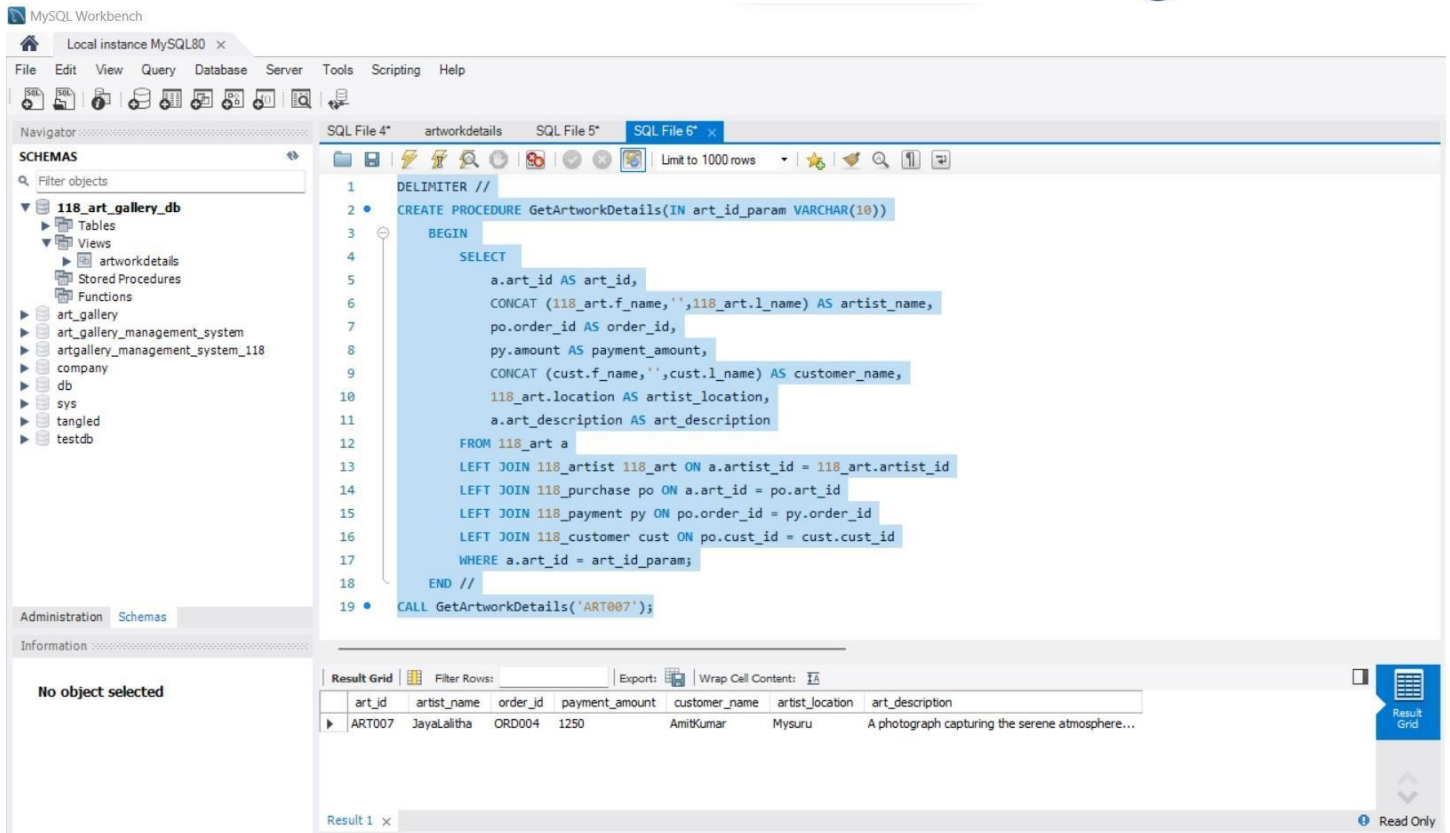
The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays the database structure for '118\_art\_gallery\_db', including tables like '118\_art', '118\_art\_order', '118\_artist', '118\_comments', and '118\_customer'. The main editor shows a SQL query in 'SQL File 5\*' that selects artwork details from the 'ArtworkDetails' table, filtering for available artworks not in the '118\_purchase' table. The 'Result Grid' at the bottom displays the query results.

art_id	art_description	artist_id	artist_name	artist_location	gallery_id	gallery_name	gallery_location	availability
ART004	A sculpture depicting the remnants of ancient a...	A003	Maresh Joshi	Mysuru	G004	Hubli Art Center	Hubli	YES
ART008	A photograph reflecting the urban landscape in ...	A006	HemaMalini	Belagavi	G007	Gulbarga Sculpture Park	Gulbarga	YES
ART010	An illustration showcasing various imaginative w...	A008	DarmendraNaik	Shivamogga	G009	Bidar Illustration Gallery	Bidar	YES
ART012	A painting that captures the delightful essence ...	A010	RaviRagavan	Dharawada	G011	Davangere Art Studio	Davangere	YES

7) In the thriving art community of your city, there's a buzz around a specific artwork with the art ID 'ART007.' This artwork has recently been purchased, and art enthusiasts are eager to know more about it. As the art gallery's database administrator, you decide to create a stored procedure to provide detailed information about this specific artwork, including the Artwork ID (art\_id), the full name of the artist (artist\_name), the Order ID (order\_id), the payment amount (payment\_amount), the full name of the customer (customer\_name), the customer's location (customer\_location), the artist's location (artist\_location), and a detailed description of the artwork (art\_description). please write an SQL query to obtain comprehensive information about the artwork with the art ID 'ART007'. Your procedure should take art\_id as input and gives respective answer and attach screenshot of procedure and output of procedure for respective art\_id.

## ANSWER:





The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with '118\_art\_gallery\_db' selected. The main editor shows a SQL query in 'SQL File 6' that creates a stored procedure 'GetArtworkDetails' and calls it with the parameter 'ART007'. The query uses a series of LEFT JOINs to retrieve artwork details, artist information, order details, payment amount, and customer information.

```

1 DELIMITER //
2 CREATE PROCEDURE GetArtworkDetails(IN art_id_param VARCHAR(10))
3 BEGIN
4     SELECT
5         a.art_id AS art_id,
6         CONCAT (118_art.f_name, ' ', 118_art.l_name) AS artist_name,
7         po.order_id AS order_id,
8         py.amount AS payment_amount,
9         CONCAT (cust.f_name, ' ', cust.l_name) AS customer_name,
10        118_art.location AS artist_location,
11        a.art_description AS art_description
12    FROM 118_art a
13    LEFT JOIN 118_artist 118_art ON a.artist_id = 118_art.artist_id
14    LEFT JOIN 118_purchase po ON a.art_id = po.art_id
15    LEFT JOIN 118_payment py ON po.order_id = py.order_id
16    LEFT JOIN 118_customer cust ON po.cust_id = cust.cust_id
17    WHERE a.art_id = art_id_param;
18 END //
19 CALL GetArtworkDetails('ART007');

```

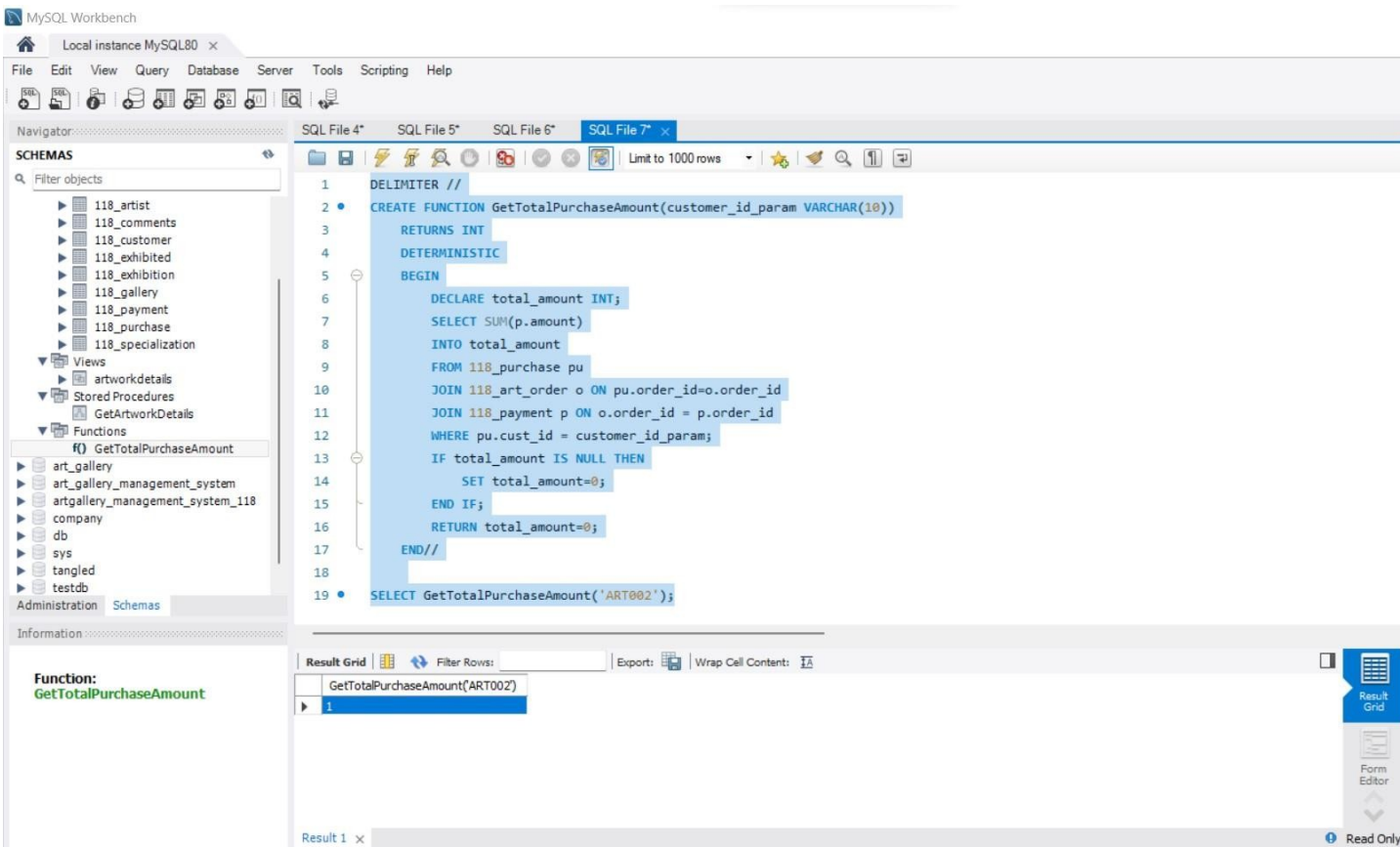
The 'Result Grid' at the bottom shows the output of the query:

art_id	artist_name	order_id	payment_amount	customer_name	artist_location	art_description
ART007	JayaLalitha	ORD004	1250	AmitKumar	Mysuru	A photograph capturing the serene atmosphere...

8) In the world of online art purchases, there's a loyal customer named Emily. She has been collecting various artworks from different artists. Emily is curious to know her total spent on art purchases from our gallery. She wants to find out the total purchase amount she has spent over the years. Using the 'GetTotalPurchaseAmount' function, please help Emily retrieve this information by providing a query that takes her customer ID as input and returns the total amount she has spent on purchases. How much has Emily invested in building her impressive art collection? Provide the output for customer with id A002?

**ANSWER:**





The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like '118\_artist', '118\_comments', '118\_customer', '118\_exhibited', '118\_exhibition', '118\_gallery', '118\_payment', '118\_purchase', and '118\_specialization'. It also shows 'Views' (e.g., 'artworkdetails'), 'Stored Procedures' (e.g., 'GetArtworkDetails'), and 'Functions' (e.g., 'GetTotalPurchaseAmount').

The main editor window shows the SQL code for creating a function:

```

1 DELIMITER //
2 CREATE FUNCTION GetTotalPurchaseAmount(customer_id_param VARCHAR(10))
3 RETURNS INT
4 DETERMINISTIC
5 BEGIN
6     DECLARE total_amount INT;
7     SELECT SUM(p.amount)
8     INTO total_amount
9     FROM 118_purchase pu
10    JOIN 118_art_order o ON pu.order_id=o.order_id
11    JOIN 118_payment p ON o.order_id = p.order_id
12    WHERE pu.cust_id = customer_id_param;
13    IF total_amount IS NULL THEN
14        SET total_amount=0;
15    END IF;
16    RETURN total_amount=0;
17 END//
18
19 SELECT GetTotalPurchaseAmount('ART002');
```

Below the editor, the 'Result Grid' shows the output of the function call:

GetTotalPurchaseAmount(ART002)
1

The bottom status bar indicates 'Result 1' and 'Read Only'.

9) In the context of a thriving art gallery, you have been tasked with creating a database to provide insights into the purchasing habits of their valued customers. You've developed a stored procedure, 'GetCustomerPurchaseDetails,' which offers a comprehensive view of a customer's name, location, total purchase amount, and a list of art IDs they've purchased. This information is vital for the gallery's marketing team to tailor promotional offers. Can you demonstrate how to use this procedure to generate a report of customer purchases, highlighting those who have made significant art acquisitions? You need to use the function you created in previous question for total purchase amount.

**ANSWER:**

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: SQL File 4\* SQL File 5\* SQL File 6\* SQL File 7\* SQL File 8\*

**SCHEMAS**

Filter objects

118\_art\_gallery\_db

Tables

Views

artworkdetails

Stored Procedures

Functions

art\_gallery

art\_gallery\_management

artgallery\_management\_s

company

db

sys

tangled

testdb

```

25 FROM 118_purchase AS pur
26 WHERE pur.cust_id = customer_id
27 );
28 INSERT INTO CustomerPurchaseDetails (customer_id, customer_name, customer_location, total_purchase_amount, art_ids)
29 VALUES (customer_id, customer_name, customer_location, total_purchase_amount, art_id_list);
30 IF done = 1 THEN
31 LEAVE customer_loop;
32 END IF;
33 END LOOP;
34 CLOSE customer_cursor;
35 SELECT * FROM CustomerPurchaseDetails;
36 DROP TEMPORARY TABLE IF EXISTS CustomerPurchaseDetails;
37 END;
38 //
39 DELIMITER ;
40 CALL GetCustomerPurchaseDetails();
41

```

Administration Schemas

Information

View: artworkdetails

Columns:

art\_id varc

art\_description varc

artist\_id varc

gallery\_id varc

availability varc

artist\_name varc

location varc

gallery\_name varc

customer_id	customer_name	customer_location	total_purchase_amount	art_ids
C002	AmitKumar	Bangaluru	0	ART001, ART007, ART013, ART006
C003	RajeshSharma	Mysuru	0	ART003
C004	PriyaPatel	Hubli	0	ART005
C005	SnehaSingh	Belgaum	1	NULL
C006	RameshGovda	Mangaluru	0	ART009
C007	KavitaReddy	Gulbarga	0	ART011
C008	ArjunRaj	Bidar	1	NULL
C009	AnitaNaidu	Udupi	0	ART002
C010	VijayKulkarni	Davangere	1	NULL
C010	VivavKulkarni	Davanore	1	NULL

Result 1 x

Read Only

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: SQL File 4\* SQL File 5\* SQL File 6\* SQL File 7\* SQL File 8\*

**SCHEMAS**

Filter objects

118\_art\_gallery\_db

Tables

Views

artworkdetails

Stored Procedures

Functions

art\_gallery

art\_gallery\_management

artgallery\_management\_s

company

db

sys

tangled

testdb

```

1 DELIMITER //
2 CREATE PROCEDURE GetCustomerPurchaseDetails()
3 BEGIN
4 declare done INT DEFAULT 0;
5 declare customer_id VARCHAR(10);
6 declare customer_name VARCHAR(100);
7 declare customer_location VARCHAR(50);
8 declare total_purchase_amount INT;
9 declare art_id_list VARCHAR(500);
10 declare customer_cursor CURSOR FOR SELECT cust_id, CONCAT(f_name, ' ', l_name) AS full_name, location
11 FROM 118_customer;
12 declare CONTINUE HANDLER FOR NOT FOUND SET done=1;
13 CREATE TEMPORARY TABLE IF NOT EXISTS CustomerPurchaseDetails(
14 customer_id VARCHAR(10),
15 customer_name VARCHAR(100),
16 customer_location VARCHAR(50),
17 total_purchase_amount INT,
18 art_ids VARCHAR(500));
19 OPEN customer_cursor;
20 customer_loop: LOOP
21 FETCH customer_cursor INTO customer_id, customer_name, customer_location;

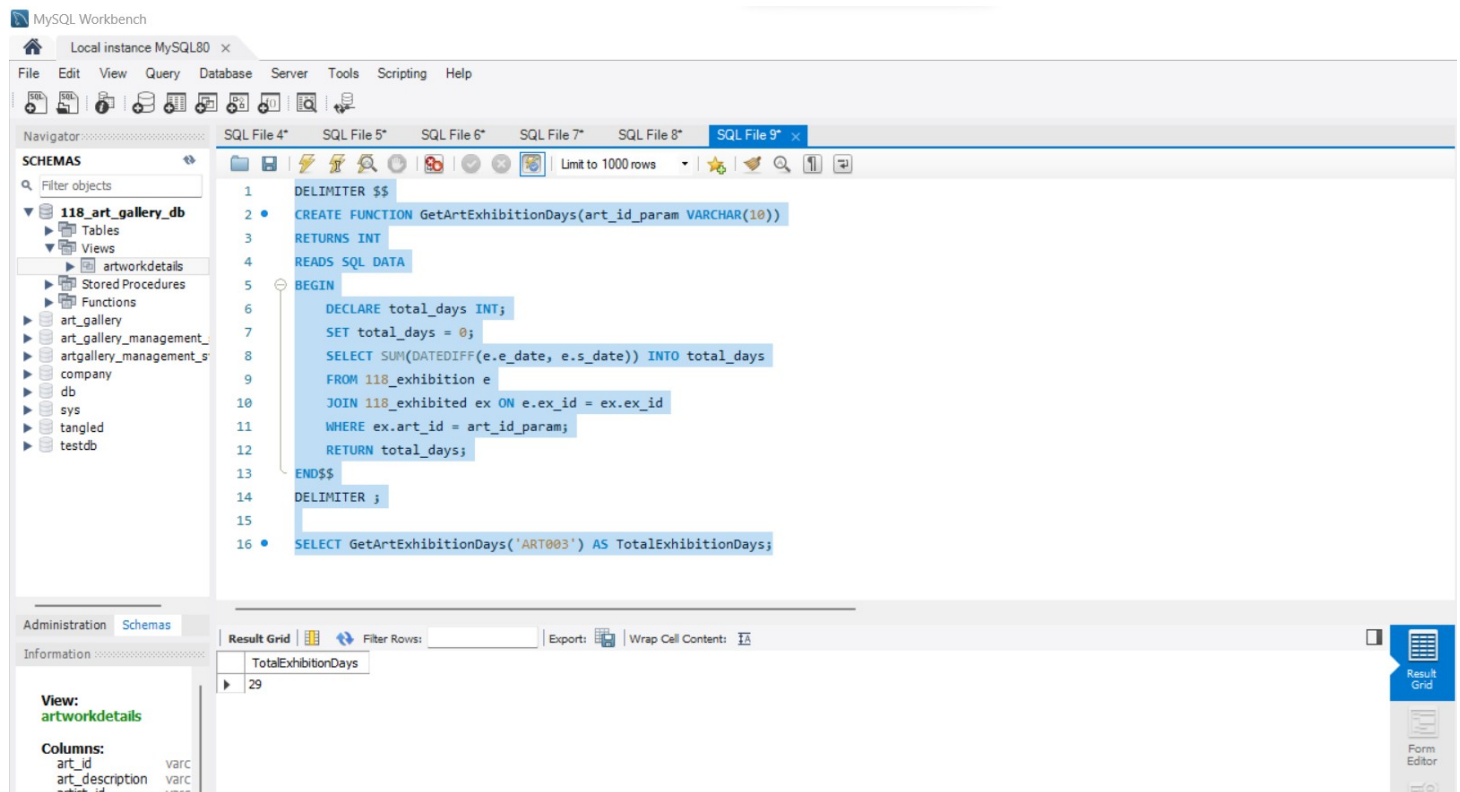
```

Administration Schemas

Information

10) In the dynamic world of art appreciation, the gallery management is keen on recognizing the impact of each artwork. They want to understand the cumulative number of days an artwork has spent being showcased across various exhibitions. For this purpose, you've been assigned the task of creating a function. This function, named `GetArtExhibitionDays`, takes an artwork ID as input and returns the total number of days the artwork has been exhibited. Execute the function for 'ART003' to unveil the intriguing story of its exhibition journey.

## ANSWER:



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with '118\_art\_gallery\_db' selected. The main editor window shows the following SQL code:

```
1 DELIMITER $$
2 CREATE FUNCTION GetArtExhibitionDays(art_id_param VARCHAR(10))
3 RETURNS INT
4 READS SQL DATA
5 BEGIN
6     DECLARE total_days INT;
7     SET total_days = 0;
8     SELECT SUM(DATEDIFF(e.e_date, e.s_date)) INTO total_days
9     FROM 118_exhibition e
10    JOIN 118_exhibited ex ON e.ex_id = ex.ex_id
11    WHERE ex.art_id = art_id_param;
12    RETURN total_days;
13 END$$
14 DELIMITER ;
15
16 SELECT GetArtExhibitionDays('ART003') AS TotalExhibitionDays;
```

The bottom panel shows the 'Result Grid' with the following data:

TotalExhibitionDays
29

Lab-5 exercise is concluded