

CS 500 Database Assignment 1

1. What relation is the result of the following queries (show your answer for each part):

A. $\sigma_{ID_PROD_TYPE < 5}$ (RefProductTypes)

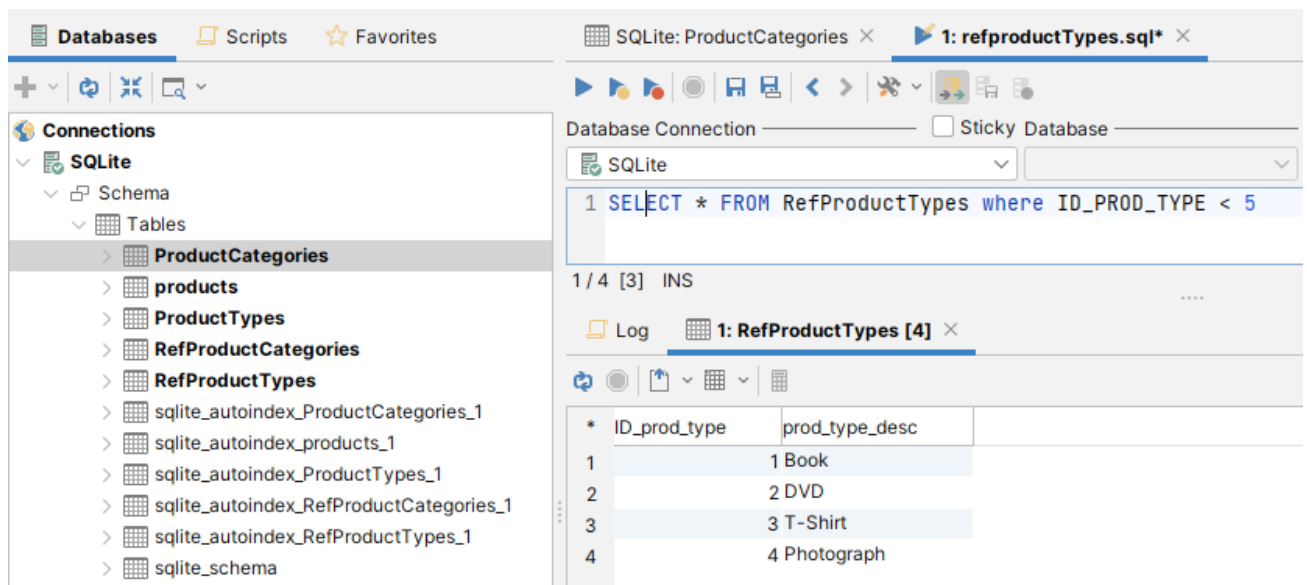
Answer:

It is a selection statement in SQL. It selects all the rows from the table RefProductTypes where the value in the ID_PROD_TYPE column is less than 5. The symbol σ represents the selection operator in SQL.

```
SELECT * FROM RefProductTypes where ID_PROD_TYPE < 5
```

The execution of the above SQL statement give four records having id_prod_type < 5

Result:-



The screenshot shows a SQLite database interface with a tree view on the left containing 'ProductCategories', 'products', 'ProductTypes', 'RefProductCategories', 'RefProductTypes', and several autoindex tables. The main window displays the SQL query: `1 SELECT * FROM RefProductTypes where ID_PROD_TYPE < 5`. Below the query, it indicates '1 / 4 [3] INS'. A log entry shows '1: RefProductTypes [4]'. The result set is displayed as a table with two columns: 'ID_prod_type' and 'prod_type_desc'.

ID_prod_type	prod_type_desc
1	1 Book
2	2 DVD
3	3 T-Shirt
4	4 Photograph

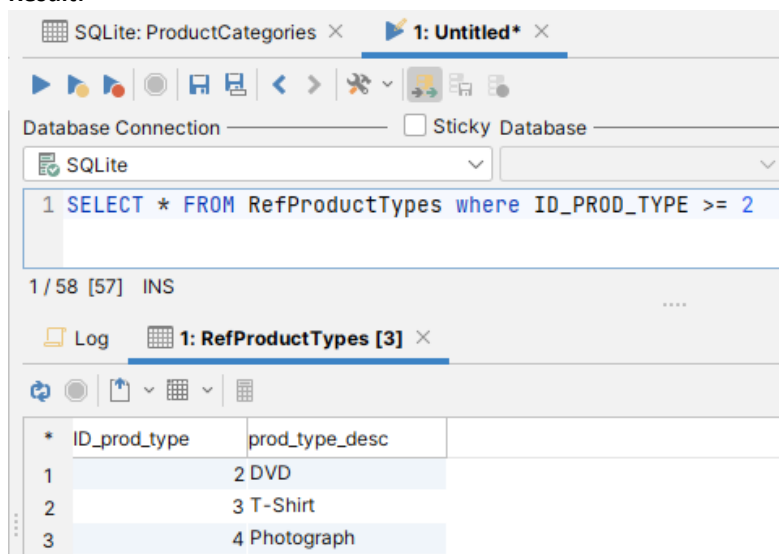
B. $\sigma_{ID_PROD_TYPE \geq 2}$ (RefProductTypes)

The statement to select products where ID_PROD_TYPE ≥ 2

```
SELECT * FROM RefProductTypes where ID_PROD_TYPE  $\geq 2$ 
```

The result shows three rows that matches the constraint.

Result:-



The screenshot shows the same SQLite database interface. The main window displays the SQL query: `1 SELECT * FROM RefProductTypes where ID_PROD_TYPE ≥ 2` . Below the query, it indicates '1 / 58 [57] INS'. A log entry shows '1: RefProductTypes [3]'. The result set is displayed as a table with two columns: 'ID_prod_type' and 'prod_type_desc'.

ID_prod_type	prod_type_desc
1	2 DVD
2	3 T-Shirt
3	4 Photograph

C. $\Pi(ID_PRODUCT((\sigma ID_PRODUCT \geq 2 \text{ and } ID_PRODUCT < 5 (Products)))) \cup \Pi(ID_PRODUCT (\sigma ID_PRODUCT > 6 (Products)))$

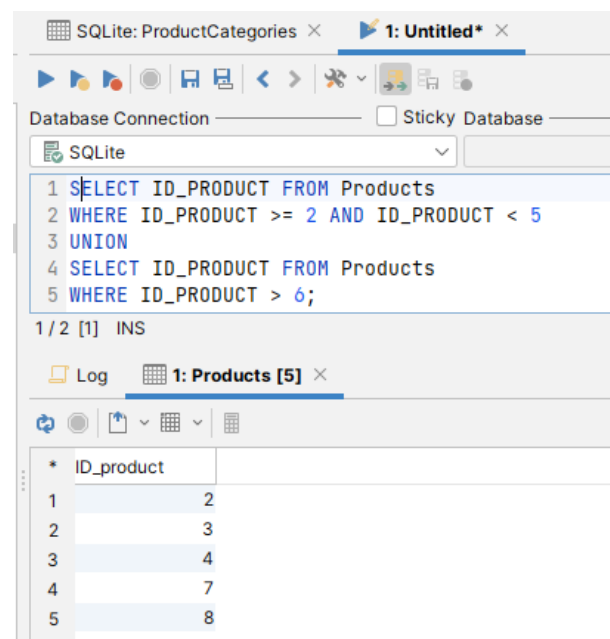
Answer:-

1. $\sigma ID_PRODUCT \geq 2 \text{ and } ID_PRODUCT < 5 (Products)$: This operation selects all rows from the Products table where the ID_PRODUCT value is greater than or equal to 2 and less than 5.
2. $\Pi(ID_PRODUCT (\sigma ID_PRODUCT > 6 (Products)))$: This operation first applies the selection (σ) to the Products table, selecting all rows where the ID_PRODUCT value is greater than 6. Then, it performs projection (Π) to keep only the ID_PRODUCT column.
3. Afterwards, take the **union (U)** of the results from steps 1 and 2.

The corresponding SQL is given below:-

```
SELECT ID_PRODUCT FROM Products
WHERE ID_PRODUCT >= 2 AND ID_PRODUCT < 5
UNION
SELECT ID_PRODUCT FROM Products
WHERE ID_PRODUCT > 6;
```

Result:-



The screenshot shows the SQLite ProductCategories application. The SQL query entered is:


```
1 SELECT ID_PRODUCT FROM Products
2 WHERE ID_PRODUCT >= 2 AND ID_PRODUCT < 5
3 UNION
4 SELECT ID_PRODUCT FROM Products
5 WHERE ID_PRODUCT > 6;
```

 The result is displayed in a table with 5 rows and 1 column (ID_product):

ID_product
2
3
4
7
8

 The interface also shows a 'Log' tab and a '1: Products [5]' tab.

D. $\Pi(ID_PRODUCT((\sigma ID_PRODUCT \geq 2 \text{ and } ID_PRODUCT < 5 (Products)))) \cap \Pi(ID_PRODUCT (\sigma ID_PRODUCT > 6 (Products)))$

Answer:-

1. $\sigma ID_PRODUCT \geq 2 \text{ and } ID_PRODUCT < 5 (Products)$: This operation selects all rows from the Products table where the ID_PRODUCT value is greater than or equal to 2 and less than 5.
2. $\Pi(ID_PRODUCT (\sigma ID_PRODUCT > 6 (Products)))$: This operation first applies the selection (σ) to the Products table, selecting all rows where the ID_PRODUCT value is greater than 6. Then, it performs projection (Π) to keep only the ID_PRODUCT column.
3. Then, take the **intersection (\cap)** of the results from steps 1 and 2.

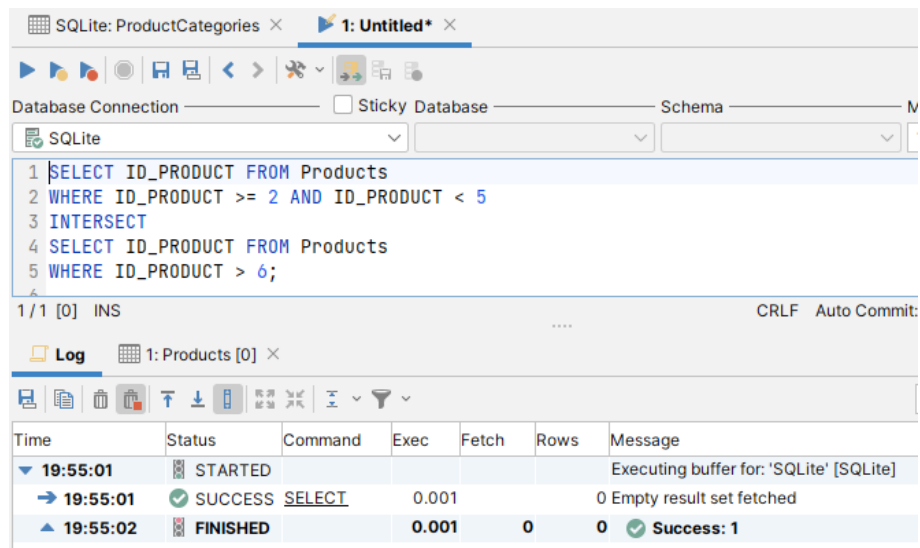
The corresponding SQL is given below:-

```
SELECT ID_PRODUCT FROM Products
WHERE ID_PRODUCT >= 2 AND ID_PRODUCT < 5
INTERSECT
SELECT ID_PRODUCT FROM Products
WHERE ID_PRODUCT > 6;
```

OR

```
SELECT ID_PRODUCT
FROM ( SELECT ID_PRODUCT
      FROM Products
      WHERE ID_PRODUCT >= 2 AND ID_PRODUCT < 5
    ) AS Subquery1
WHERE ID_PRODUCT > 6;
```

Result :- NO rows returned,



E. $(ProductTypes) \cap (RefProductTypes)$ (10 points)

Answer:-

The Intersection simply filter the rows that are present in both the tables **ProductTypes** as well as in **RefProductTypes**.

The intersection of the **ProductTypes** and **RefProductTypes** relations can be expressed in SQL using the INNER JOIN operation. SQL statement is given below

```
SELECT * FROM ProductTypes
```

```
INNER JOIN RefProductTypes
```

```
ON ProductTypes.ID_PROD_TYPE = RefProductTypes.ID_PROD_TYPE;
```

Result:-

The screenshot shows a SQLite database interface with a query window titled '1: Untitled*'. The query is:

```
1 SELECT * FROM ProductTypes
2 INNER JOIN RefProductTypes
3 ON ProductTypes.ID_PROD_TYPE = RefProductTypes.ID_PROD_TYPE;
```

The results are displayed in a table with 8 rows and 5 columns. The columns are: ID_PRODUCT, ID_PROD_TYPE, ID_prod_type, prod_type_desc, and an unnamed column. The data is as follows:

	ID_PRODUCT	ID_PROD_TYPE	ID_prod_type	prod_type_desc	
1	1	1	1	1 Book	
2	2	1	1	1 Book	
3	3	3	3	3 T-Shirt	
4	4	3	3	3 T-Shirt	
5	5	1	1	1 Book	
6	6	4	4	4 Photograph	
7	7	2	2	2 DVD	
8	8	4	4	4 Photograph	

2. What relation is the result of the following queries (show your answer for each part): (10 Points)

Answer:-

A. $\Pi_{PRODUCT_NAME, CATEGORY_DESC}(Products \bowtie ProductCategories \bowtie RefProductCategories)$:

This is to retrieve the product names and their corresponding category descriptions where the product exists in both ProductCategories and RefProductCategories, INNER JOIN operation can work here. SQL statement:

SELECT

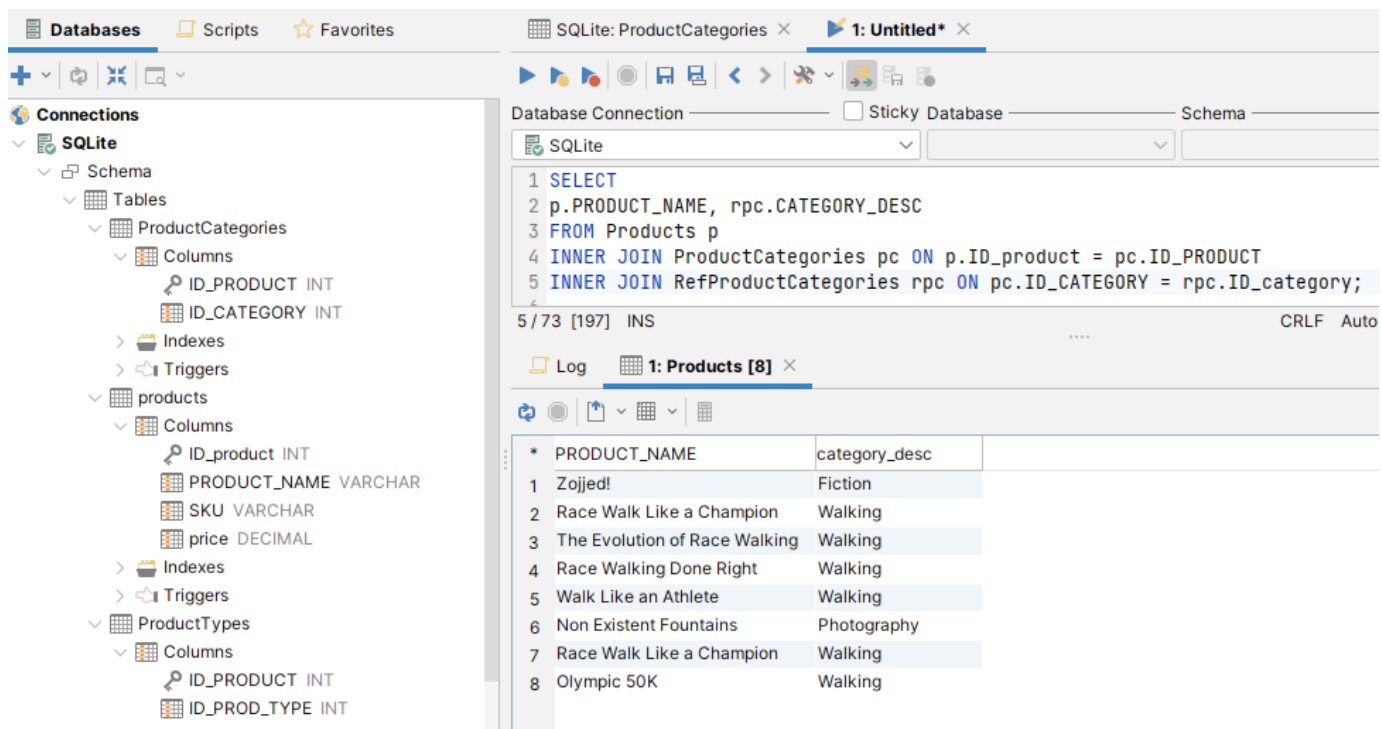
p.PRODUCT_NAME, rpc.CATEGORY_DESC

FROM Products p

INNER JOIN ProductCategories pc ON p.ID_product = pc.ID_PRODUCT

INNER JOIN RefProductCategories rpc ON pc.ID_CATEGORY = rpc.ID_category;

Result:-



B. PRODUCT_NAME, CATEGORY_DESC(Products ROJ ProductCategories ROJ RefProductCategories) (10 Points)

Answer:-

RIGHT OUTER JOIN operation used here on Products, ProductCategories, and RefProductCategories tables. Give all the categories present in RefProductsategories ie. The Right Table and the matching rows from the left tables.

SELECT Products.PRODUCT_NAME, RefProductCategories.CATEGORY_DESC

FROM Products

RIGHT OUTER JOIN

ProductCategories ON Products.ID_PRODUCT = ProductCategories.ID_PRODUCT

RIGHT OUTER

JOIN RefProductCategories ON ProductCategories.ID_CATEGORY = RefProductCategories.ID_CATEGORY

Result:-

The screenshot shows a SQLite database interface with a query window and a results window. The query window contains the following SQL query:

```

1 SELECT Products.PRODUCT_NAME, RefProductCategories.CATEGORY_DESC |
2 FROM Products
3 RIGHT OUTER JOIN ProductCategories ON Products.ID_PRODUCT = ProductCategories.ID_PRODUCT
4 RIGHT OUTER JOIN RefProductCategories ON ProductCategories.ID_CATEGORY = RefProductCategories.ID_CATEGORY

```

The results window shows the following data:

PRODUCT_NAME	category_desc
Zojjed!	Fiction
Race Walk Like a Champion	Walking
The Evolution of Race Walking	Walking
Race Walking Done Right	Walking
Walk Like an Athlete	Walking
Non Existent Fountains	Photography
Race Walk Like a Champion	Walking
Olympic 50K	Walking
(null)	Computer Science

3. Write a relational algebra query to return the SKUS of all Products that have a product category of "Fiction" and a Product type of "DVD". (10 points)

Answer:-

The below algebra uses the selection operator σ to filter the Products relation based on the conditions that the CATEGORY_DESC equal to "Fiction" and the PRODUCT_TYPE equal to "DVD". The resulting relation contains only the SKUS of the products that meet these conditions due to projection on SKUS.

$\Pi(\text{SKUS} (\sigma \text{ CATEGORY_DESC} = \text{'Fiction'} \wedge \text{PRODUCT_TYPE} = \text{'DVD'} (\text{Products})))$

4. Write the query, in relational algebra, to get the names of all products that do not have any product types assigned to them. (You may use the joins we discussed in class) (10 points)

Answer:-

Here we join Products and ProductType on common attribute 'ID_Product', Next we subtract all products, which will give us the remaining products that are not assigned and type, then we project on NAME.

$\Pi \text{ name} (\text{Products} - (\text{Products} \bowtie \text{ProductTypes}))$

5. Write the query, in relational algebra, to get the name of the lowest id_category in the RefProductCategories table (You may use the joins in class, but not the aggregate functions) (questions 6-8, you may NOT use the join conditions learned in class) (i.e. inner, natural, left, right, and full) (10 points)

Answer:-

The below query is a nested relational expression without aggregation function like MIN, to get the names of lowest ID_Category in RefProductCategories

$\Pi \text{ name} (\sigma \text{ NOT EXISTS} (\text{RefProductCategories AS R2 WHERE R2.id_category} < \text{RefProductCategories.id_category}) (\text{RefProductCategories}))$

6. Write a query in relational algebra to return product names of all products with no categories assigned to them. (10 points)

Answer:-

Difference operator is used to subtract the names of products that have a category from the names of all products and Π operator is used to project the PRODUCT_NAME attribute, x stands for product.

Π PRODUCT_NAME (Products) - Π PRODUCT_NAME (ProductCategories) \times Π PRODUCT_ID (Products)

7. Write the query, in relational algebra, to names of the products that have a category of “Wedding” and also have a product type of “Gown”. (10 points)

Π PRODUCT_NAME (σ CATEGORY_DESC = 'Wedding' \wedge PRODUCT_TYPE = 'Gown' (Products))

8. Write the query, in relational algebra, to get a complete list of products (NAME, SKU, PRICE), and the descriptions of all product types associated with them. (10 points)

Answer:- SQL is

```
SELECT p.price, p.product_name, p.sku, rpt.prod_type_desc
FROM products as p, refproducttypes as rpt, producttypes as pt
WHERE p.ID_product = pt.ID_PRODUCT
AND pt.id_prod_type = rpt.ID_prod_type
```

Relational Algebra is

Π (p.PRODUCT_NAME, p.SKU, p.PRICE, rpt.PROD_TYPE_DESC)
((PRODUCTS p * PRODUCTTYPES pt)
 σ p.ID_PRODUCT = pt.ID_PRODUCT \wedge PT.ID_PROD_TYPE = RPT.ID_PROD_TYPE)

9. Write a query, in relational algebra, to return the name of products that have Product Types but not Product Categories

Answer:-

1. Find the products that have product types.
 $T1 = \Pi(\text{PRODUCT_NAME})(\sigma(\text{TYPE_ID} \neq \text{NULL})(\text{Products}))$
2. Find the products that have product categories.
 $T2 = \Pi(\text{PRODUCT_NAME})(\sigma(\text{CATEGORY_ID} \neq \text{NULL})(\text{Products}))$
3. Now find the difference between T1 and T2 to get the products with product types but not product categories:
Result = T1 - T2
The resulting relation will contain the names of products that fulfill these criteria.