

MINI PROJECT REPORT – DBMS I & II

24-06-2023



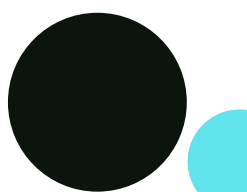
Group -03:


Jamima Yeasmin
Kowsalya S
Harsh Chaudhary
Nandita Malakar
Niraj Shah




What problem are we solving?

A. Retail Banking Case Study:

1. The case study addresses a significant challenge associated with the efficient management and querying of data in a retail banking system. It encompasses a set of specific tasks that collectively contribute to the resolution of this challenge in a meticulous and comprehensive manner.
 2. The initial task involves the creation of a well-structured and purposeful database schema tailored to the requirements of the retail banking system. This intricate process requires careful consideration of various entities including branches, employees, customers, accounts, and transactions. By designing a robust database schema, the case study aims to establish a solid foundation for effectively managing the extensive range of data within the retail banking system.
 3. Another crucial aspect of the case study involves the formulation of SQL queries. These queries are specifically designed to retrieve and manipulate data from the database.
 4. By employing sophisticated SQL techniques, the case study aims to facilitate seamless access to relevant information and enable efficient data manipulation operations. The ultimate objective is to extract
- 



meaningful insights and optimize decision-making processes within the retail banking system.

5. Furthermore, the case study emphasizes the implementation of triggers, which serve as vital mechanisms for automating processes within the database. In this context, triggers are employed to update balances in the account table based on transactions recorded in the transaction table. By utilizing triggers, the case study aims to ensure accurate and real-time updates to account balances, enhancing the reliability and integrity of financial data within the retail banking system.
 6. Overall, the case study's primary goal is to establish effective data management practices and enable effortless retrieval of critical information pertaining to branches, employees, customers, accounts, and transactions within the retail banking system. By addressing these specific tasks, the case study endeavors to enhance operational efficiency, streamline decision-making processes, and promote data-driven insights within the retail banking domain.
- 



B. E – Commerce Retail Product Database Analysis:

1. Product Details and Sorting:


The system needs to display the details of products based on specific criteria and sort them in descending order of their category. Additionally, the system needs to adjust the prices of certain categories by specified amounts.

2. Shipped Product Details:

The system needs to list the description, class description, and price of all products that have been shipped. This helps in analyzing the characteristics and pricing of products that have already been sent to customers.

3. Inventory Status:

The system needs to determine the inventory status of different product categories based on their available quantities. This involves classifying the inventory as "Low stock," "In stock," or "Enough stock" according to predefined quantity ranges for each category. In case the available quantity is zero, it should be labeled as "Out of stock."





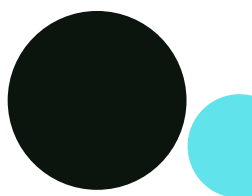
4. Customer Segmentation based on Location and Purchase Behavior:


The system needs to identify customers located outside Karnataka who have not purchased any toys or books. This allows for segmenting customers based on their geographical location and specific purchase patterns, enabling targeted marketing or further analysis.

How did you solve the problem?


A. Retail Banking Case Study:

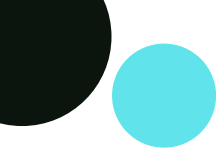

- i. The system needs to identify customers located outside Karnataka who have not purchased any toys or books. This allows for segmenting customers based on their geographical location and specific purchase patterns, enabling targeted marketing or further analysis.
- ii. Leveraging the comprehensive ER Diagram, we skillfully discerned the interconnections shared by the diverse tables within the database. By thoroughly examining the relationships, we were able to gain a profound understanding of how each table relates to one another, thus facilitating a holistic comprehension of the database structure.



- 
- iii. This meticulous analysis proved instrumental in providing us with a comprehensive understanding of the database's table composition. Armed with this knowledge, we were able to formulate a well-defined approach for extracting meaningful and desired insights from the available data. After carefully mapping out our strategy, we executed the pertinent queries, successfully retrieving valuable information that met our intended objectives.

B. E-Commerce Retail Product Database Analysis:

- i. In this project, we started by creating a database called "Case-Study" and proceeded to create all the necessary entities based on the given instructions. We carefully followed the provided guidelines to ensure the correct structure of the database.
 - ii. After setting up the database and creating the required tables, we examined the Entity-Relationship (ER) Diagram. Our goal was to identify the common connecting points and examine the relationships between the various tables within the database.
- 

- 
- iii. During our analysis of the ER Diagram, we observed that there were no pre-established relationships between the entities. Therefore, we took the initiative to establish the necessary relationships within the ER Diagram. This step was crucial to facilitate a better understanding of the data and to derive meaningful insights from it.
 - iv. Once we had a clear understanding of the relationships between the entities, we proceeded to analyze the given questions. We carefully determined the appropriate approach and methodology for solving each question, considering the structure and relationships within the database.
 - v. Finally, we successfully solved the questions and obtained the desired output. Our approach involved utilizing the established relationships, querying the database, performing calculations, and generating the required results.
 - vi. Throughout this process, we ensured adherence to the instructions, meticulously examined the database structure, and leveraged the relationships established in the ER Diagram to provide accurate and relevant solutions to the questions.
- 

What were the problems you took to solve the problem?

- **Step1:** We created the Database as per the conditions provided to us and assigned the constraints.
- **Step2:** We inserted the values and generated the ER diagram.

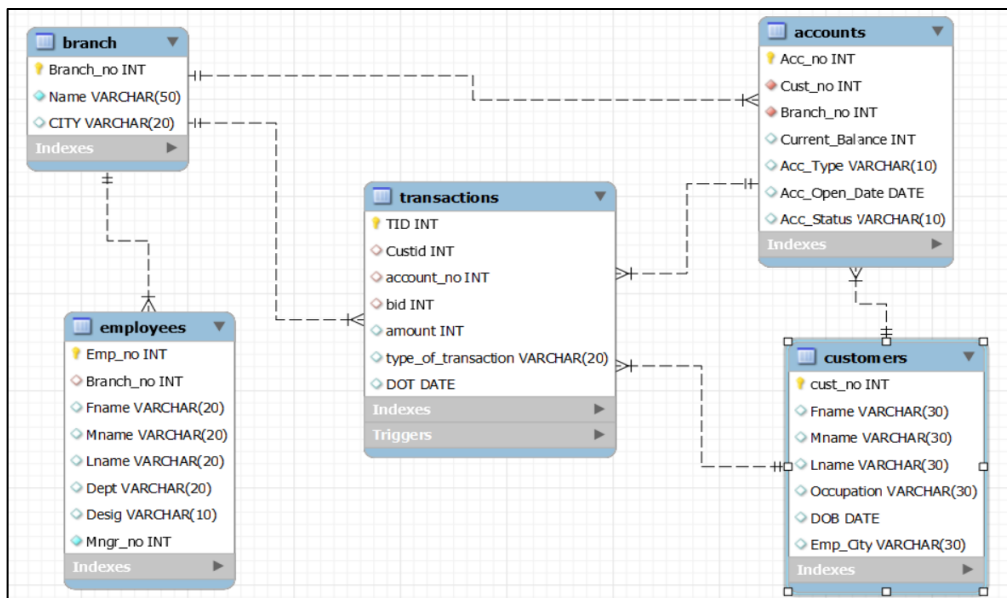


Fig 1: Entity relationship (ER) diagram of the A. Retail Banking Case Study.

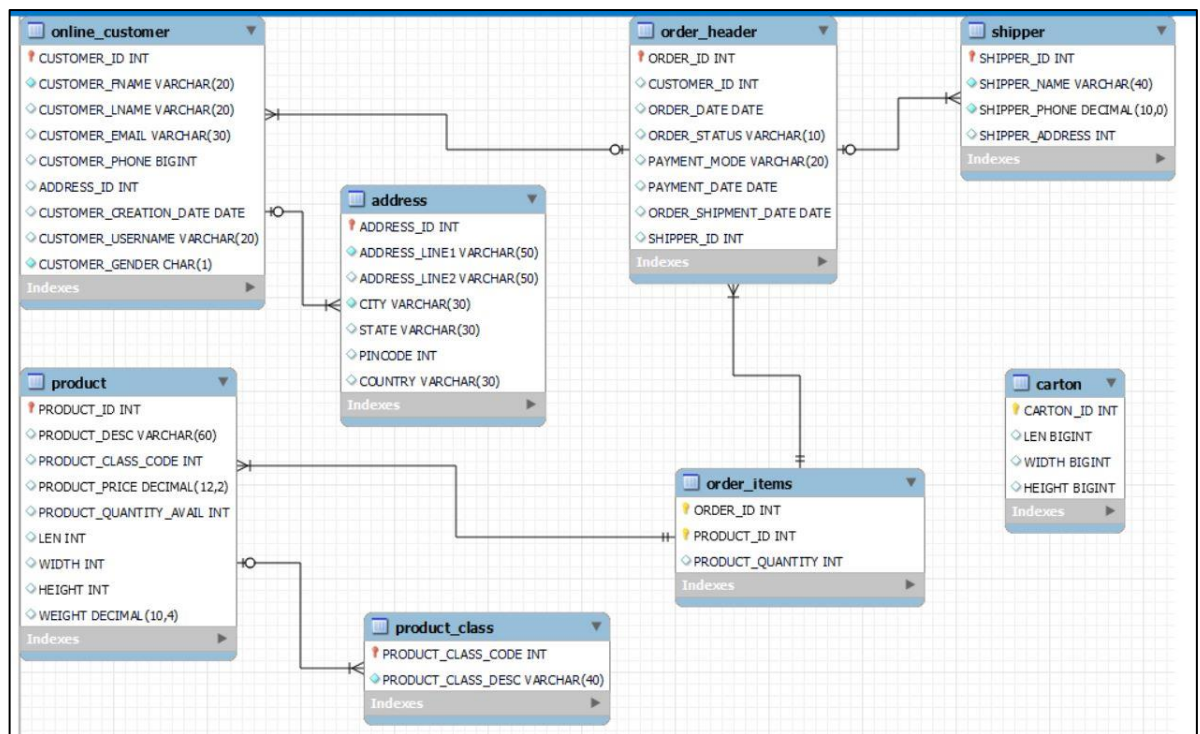


Fig 2: Entity relationship (ER) diagram of the B. E-Commerce Retail Product Database Analysis.

- **Step3:** We analyzed the ER diagram, and we thus studied the relationship between the different entities in the Database and noted the common attributes between the different tables for further references.
- **Step4:** Verified the ER Diagram and ensured the datatype of all the attributes in different entities to ensure if any altering of attribute is required.
- **Step5:** We checked all the entities to ensure all data was present for smooth retrieving of the meaningful information. Also, we noticed that some data for an attribute city from the entity 'customers' was missing.

We requested help on the clarification of missing values of attribute “city” from “customers” table.

- **Step6:** Proceeded with the questions.
- **Step7:** Found out the unique occupation from the Customers table using “Distinct” command and gain insights on “Accounts” table after sorting the values in Ascending order.

```
# 3. Select unique occupation from customer table :  
select distinct Occupation from customers;
```

```
# 4. Sort accounts according to current balance :  
select * from accounts order by Current_Balance;
```

Occupation
Service
Business

	Acc_no	Cust_no	Branch_no	Current_Balance	Acc_Type	Acc_Open_Date	Acc_Status
▶	2	2	2	5000	Savings	2012-06-12	Active
	1	1	1	10000	Savings	2012-12-15	Active
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig 3: Queries and output for unique occupation and account tables post sorting.

- **Step8:** We found out the DOB of a customer from the “customers” table and added a new column “city” to “Branch” Table.

```
# 5. Find the Date of Birth of customer name 'Ramesh' :
select DOB from customers where Fname = "Ramesh";

# 6. Add column city to branch table :
alter table branch add column CITY varchar(20);
SELECT * FROM bank_project.branch;
```

	Fname	DOB
▶	Ramesh	1976-12-06

Branch_no	Name	CITY
1	Delhi	NULL
2	Mumbai	NULL
NULL	NULL	NULL

Fig 4: Queries and output pertaining to the DOB of customer and addition of the "city" column to "branch" table.

- **Step9:** Updated Mname & Lname.

Also fetched the information from "accounts" table by finding account open date between provided dates using "between" conditions.

```
# 7. Update the mname and lname of employee 'Bella' and set to 'Karan', 'Singh'
update employees set Mname = "Karan", Lname = "Singh" where Fname = "Bella";
SELECT * FROM bank_project.employees;

# 8. Select accounts opened between '2012-07-01' AND '2013-01-01'
select * from accounts where Acc_Open_Date between '2012-07-01' AND '2013-01-01';
```

Emp_no	Branch_no	Fname	Mname	Lname	Dept	Desig	Mngr_no
1	1	Mark	Steve	Lara	Account	Accountant	2
2	2	Bella	Karan	Singh	Loan	Manager	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Acc_no	Cust_no	Branch_no	Current_Balance	Acc_Type	Acc_Open_Date	Acc_Status
1	1	1	10000	Savings	2012-12-15	Active
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig 5: Queries and Output for updating the mname and lname of a customer and retrieving account details within a specific date.

- **Step10:** Found out the customers whose name has 2nd letter as “a”, for this we used “Wild card” characters like ‘% and _’.

Connected the customers table and accounts table using “INNER - JOIN” and found details of customer who had the lowest account balance by using “ORDER BY” and used “LIMIT 1” for the desired output.

```
# 9. List the names of customers having 'a' as the second letter in their names :  
select Fname from customers where Fname like "_a%";  
  
# 10. Find the lowest balance from customer and account table :  
select c.cust_no, a.Acc_no, c.Fname, c.Mname, c.Lname, a.Current_Balance from customers c  
join accounts a on c.cust_no = a.Cust_no  
order by a.Current_Balance  
limit 1;
```

	Fname
►	Ramesh

cust_no	Acc_no	Fname	Mname	Lname	Current_Balance
2	2	Avinash	Sunder	Minha	5000

Fig 6: Queries and Output for retrieving customer whose name contains “a” as second character and identifying the customer with the lowest account balance.

- **Step11:** To determine the customer count for each occupation category, we initially selected the occupation column and applied the count function on the cust_no column in the customers table. Then, we used the group feature to group the results based on occupation. Retrieved the first and last name of employees and applied a where condition to filter out employees whose designation was “Manager” from the employees table.

```
-- 11. Give the count of customer for each occupation
SELECT occupation,
       Count(cust_no) AS count_of_customer
FROM   customers
GROUP BY occupation;

-- 12. Write a query to find the name (first_name, last_name) of the employees who are managers.
SELECT fname,
       lname
FROM   employees
WHERE  desig = 'Manager';
```

	occupation	count_of_customer		fname	lname
►	Service	1			
	Business	1	►	Bella	Singh

Fig 7: Queries and Output for retrieving count of customer in each occupation sector and get first and last name of employees who are Managers.

- **Step12:** To retrieve information about employees whose names ends with 'a', we initially extracted the fname, mname and lname fields from the employees table. We then used a WHERE clause along with the '%' wildcard character to filter the retrieved results appropriately.

We retrieved employee information for individuals associated with the 'credit' or 'loan' departments. This was achieved by selecting relevant information from the employees table and applying a filter using the WHERE condition to match the department values to 'credit' or 'loan'.

```
-- 13. List name of all employees whose name ends with a
SELECT fname,
       mname,
       lname
FROM   employees
WHERE  lname LIKE '%a';

-- 14. Select the details of the employee who work either for department 'loan' or 'credit'
SELECT *
FROM   employees
WHERE  dept LIKE 'Loan'
       OR 'Credit';
```

	fname	mname	lname
▶	Mark	Steve	Lara

	emp_no	branch_no	fname	mname	lname	dept	desig	mngnr_no
▶	2	2	Bella	Karan	Singh	Loan	Manager	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig 8: Queries and Output for retrieving employees whose name ends with a and details of employees whose department are loan or credit.

- **Step13:** We retrieved the cust_no, fname and dob from the customer table, as well as the acc_no from the accounts table, by performing an inner join. Then we applied a filter using a WHERE clause to narrow down the results based on a date greater than 15.

```
-- 15. Write a query to display the customer number, customer firstname, account number for the customer's who are born after 15th of any month.  
SELECT c.cust_no,  
       c.fname,  
       a.acc_no,  
       c.dob  
FROM   customers c  
       JOIN accounts a  
       ON c.cust_no = a.cust_no  
WHERE  Day(dob) > 15;
```

	cust_no	fname	acc_no	dob
►	2	Avinash	2	1974-10-16

Fig 9: Queries and Output for retrieving customer details and account numbers for customers with a birth date greater than the 15th day of any month.

- **Step14:** We obtained the cust_no and fname from the customers table, branch_no from the branch table, and current_balance from the accounts table. We utilized an inner join to establish connections between the tables to obtain the desired output.

```
-- 16. Write a query to display the customer's number, customer's firstname, branch id and balance amount for people using JOIN.  
SELECT c.cust_no,  
       c.fname,  
       b.branch_no,  
       a.current_balance  
FROM   customers c  
       JOIN accounts a  
       ON c.cust_no = a.cust_no  
       JOIN branch b  
       ON a.branch_no = b.branch_no;
```

	cust_no	fname	branch_no	current_balance
►	1	Ramesh	1	10000
	2	Avinash	2	5000

Fig 10: Queries and Output for retrieving customer details, current balance and branch id and balance amount for people using JOIN.

- **Step15:** We created a view to store customer information for individuals living in the same city as their respective bank branches. This involved employing an inner join operation that encompassed the customers, branches, and accounts tables, allowing us to successfully retrieve the desired data. Given the absence of a direct link between the customer and branches table, we leveraged the accounts table as an intermediary component to establish the necessary connection.

<pre>-- 17. Create a virtual table to store the customers who are having the accounts in the same city as they live UPDATE customers SET city = 'Delhi'; CREATE VIEW city_cust AS SELECT c.cust_no, c.fname, c.mname, c.lname, c.city, c.dob, c.occupation, a.acc_no, b.branch_name FROM customers AS c JOIN accounts AS a ON a.cust_no = c.cust_no JOIN branch AS b ON b.branch_no = a.branch_no WHERE c.city = b.branch_name; SELECT * FROM city_cust;</pre>									
	cust_no	fname	mname	lname	city	dob	occupation	acc_no	branch_name
▶	1	Ramesh	Chandra	Sharma	Delhi	1976-12-06	Service	1	Delhi

Fig 11: Queries and Output for retrieving customer details, current balance and branch id and balance amount for people using JOIN.

- **Step16:** We implemented a trigger named “Update_account_balance” for the “accounts” table. The purpose of this trigger is to automatically update the “current_balance” in the “accounts” table based on the transactions inserted into the transactions table. Whenever a new row is inserted into the “transaction” table, the trigger is activated. It examines the “type_of_transaction” specified. If the transaction type is ‘deposit’, the trigger increases the current balance in the corresponding row of “accounts” table. Conversely, if the transaction type is ‘withdraw’, the trigger decreases the current balance accordingly.

```
-- a. Write trigger to update balance in account table on Deposit or Withdraw in transaction table
DELIMITER //
CREATE TRIGGER update_account_balance
After
INSERT ON TRANSACTION
FOR each row
BEGIN
    DECLARE current_balance_updated decimal (10,2);
    select current_balance
    INTO    current_balance_updated
    FROM    accounts
    WHERE   acc_no = new.account_no;

    UPDATE accounts
    SET     current_balance =
        CASE
            WHEN new.type_of_transaction = 'deposit' THEN current_balance_updated + new.amount
            WHEN new.type_of_transaction = 'withdraw' THEN current_balance_updated - new.amount
        END
    WHERE   acc_no = new.account_no;
end//
delimiter ;

-- b. Insert values in transaction table to show trigger success

INSERT INTO transaction (TID, custid, account_no, branch_id, amount, type_of_transaction, DOT)
VALUES
('1', '1', '1', '1', '3000', 'Withdraw', '2023-06-23');
```

	TID	custid	account_no	branch_id	amount	type_of_transaction	DOT
▶	1	1	1	1	3000	Withdraw	2023-06-23
✱	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig 12.a: Transaction values being inserted.

	acc_no	branch_no	cust_no	acc_type	acc_open_date	current_balance	account_status
▶	1	1	1	Savings	2012-12-15	10000	Active
✱	2	2	2	Savings	2012-06-12	5000	Active
✱	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig 12.b: Accounts table before the trigger

	acc_no	branch_no	cust_no	acc_type	acc_open_date	current_balance	account_status
▶	1	1	1	Savings	2012-12-15	7000	Active
✱	2	2	2	Savings	2012-06-12	5000	Active
✱	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig 12.b: Accounts table after trigger with updated current balance.

Fig 12: Query and output for implementing trigger to update current balance of account based on type of transaction.

- Step17:** In order to identify the customer with the second highest account balance we retrieved the necessary customer details from the “customers” table. To obtain the account balance information we utilized the inner join operation with the “accounts” table, linking the two tables based on the common “cust_no” column. To ensure the desired order, we sorted the result set by the “current_balance” column in descending order, using the “ORDER BY” clause. To get the second highest balance, we utilized the “LIMIT” clause with an offset value of 1, effectively skipping the first row (first highest balance).

```
-- 19. Write a query to display the details of customer with second highest balance

select c.cust_no, c.fname, c.mname, c.lname, a.current_balance
from customers c
join accounts a
on c.cust_no = a.cust_no
order by current_balance desc
limit 1,1;
```

	cust_no	fname	mname	lname	current_balance
▶	2	Avinash	Sunder	Minha	5000

Fig 13: Query and output for getting the details of customer with the second highest balance.

- **Step18:** To display the product details based on specific criteria and sorting them in descending order of category, the select statement was used to retrieve specific columns from the “product” table. A case statement was implemented to calculate the updated price based on the product class code. Then an order by clause was used to sort the result in descending order of the product class code.

```
-- 1. Display the product details as per the following criteria and sort them in descending order of category:
• SELECT
  product_class_code,
  product_id,
  product_desc,
  PRODUCT_PRICE,
  CASE
    WHEN product_class_code = 2050 THEN product_price + 2000
    WHEN product_class_code = 2051 THEN product_price + 500
    WHEN product_class_code = 2052 THEN product_price + 600
    ELSE product_price
  END AS price_updated
FROM product
ORDER BY product_class_code DESC;
```

product_class_code	product_id	product_desc	product_price	price_updated
3002	99992	Tom Clancy's Ghost Recon: Future Soldier (PC Game)	999	999
3002	99993	Nokia 1280 (Black)	999	999
3002	99991	Dell Targus Synergy 2.0 Backpack	999	999
3001	99996	Nokia Asha 200 (Graphite)	4070	4070
3001	99995	LG MS-2049UW Solo Microwave	4800	4800
3001	99994	HP Deskjet 2050 All-in-One - J510a Printer	3749	3749
3000	99999	Samsung Galaxy Tab 2 P3100	19300	19300
3000	99990	Quanta 4 Port USB Hub	500	500
3000	99998	Nikon Coolpix L810 Bridge	14987	14987
3000	99997	Sony Xperia U (Black White)	16499	16499

Fig 14: Query and output for displaying the product details based on specific criteria and sorting them in descending base on their product_code.

- **Step19:** To obtain the list of product descriptions, class descriptions, and process for all shipped products, the relevant columns were selected from the “product”, “product_class” and “order_header” tables. Utilizing the inner join operation to connect the “product_class” and “product” tables using the shared key value “product_class_code”. However, as the “order_header” table lacked direct links to the “product” or “product_class” table the “order_items” table was used as an intermediary.

```
-- 2. List the product description, class description and price of all products which are shipped.

select p.product_desc , p.product_price, pc.product_class_desc , oh.order_status
from product p
join product_class pc on p.product_class_code = pc.product_class_code
join order_items oi on p.product_id = oi.PRODUCT_ID
join order_header oh on oi.order_id = oh.ORDER_ID
where order_status = 'Shipped';
```

	product_desc	product_price	product_class_desc	order_status
▶	Zamark Color Pencil Art Set	100.00	Stationery	Shipped
	Women Hand Bag	1600.00	Bags	Shipped
	Women Hand Bag	1600.00	Bags	Shipped
	Women Hand Bag	1600.00	Bags	Shipped
	Women Hand Bag	1600.00	Bags	Shipped
	TRANS 2D A4 Size Box File	120.00	Stationery	Shipped

Fig 15: Query and output for getting the details of product description , product class description and price or products that are shipped.

- **Step20:** To determine the inventory status of the products based on their available quantity. We selected the relevant columns from the “product” and “product_class” tables by utilizing the inner join operation. The join was performed based on the common key “product_class_code”. After selecting the columns, we applied necessary case conditions to determine the inventory status of each product. These conditions were based on the available quantity of the products. The resulting inventory status was then displayed in a dedicated column labeled as “inventory status”.

```
-- 3. Show inventory status of products as below as per their available quantity:
#a. For Electronics and Computer categories, if available quantity is < 10, show 'Low stock', 11 < qty < 30, show 'In stock', > 31, show 'Enough stock'
#b. For Stationery and Clothes categories, if qty < 20, show 'Low stock', 21 < qty < 80, show 'In stock', > 81, show 'Enough stock'
#c. Rest of the categories, if qty < 15 - 'Low Stock', 16 < qty < 50 - 'In Stock', > 51 - 'Enough stock'
#For all categories, if available quantity is 0, show 'Out of stock'.

SELECT
  p.product_id, p.product_desc, p.PRODUCT_QUANTITY_AVAIL, pc.PRODUCT_CLASS_DESC,
  CASE
    WHEN PRODUCT_CLASS_DESC IN ('Electronics', 'Computer') AND PRODUCT_QUANTITY_AVAIL < 10 THEN 'Low stock'
    WHEN PRODUCT_CLASS_DESC IN ('Electronics', 'Computer') AND 11 < PRODUCT_QUANTITY_AVAIL < 30 THEN 'In stock'
    WHEN PRODUCT_CLASS_DESC IN ('Electronics', 'Computer') AND PRODUCT_QUANTITY_AVAIL >= 31 THEN 'Enough stock'
    WHEN PRODUCT_CLASS_DESC IN ('Stationery', 'Clothes') AND PRODUCT_QUANTITY_AVAIL < 20 THEN 'Low stock'
    WHEN PRODUCT_CLASS_DESC IN ('Stationery', 'Clothes') AND 21 < PRODUCT_QUANTITY_AVAIL < 80 THEN 'In stock'
    WHEN PRODUCT_CLASS_DESC IN ('Stationery', 'Clothes') AND PRODUCT_QUANTITY_AVAIL >= 81 THEN 'Enough stock'
    WHEN PRODUCT_QUANTITY_AVAIL < 15 THEN 'Low stock'
    WHEN PRODUCT_QUANTITY_AVAIL >= 16 AND PRODUCT_QUANTITY_AVAIL < 50 THEN 'In stock'
    WHEN PRODUCT_QUANTITY_AVAIL >= 51 THEN 'Enough stock'
    ELSE 'Out of stock'
  END AS inventory_status
FROM product p
join product_class pc on p.PRODUCT_CLASS_CODE = pc.PRODUCT_CLASS_CODE;

#3b. For Stationery and Clothes categories, if qty < 20, show 'Low stock', 21 < qty < 80, show 'In stock', > 81, show 'Enough stock'
SELECT p.PRODUCT_DESC AS "Product Name", pc.PRODUCT_CLASS_DESC AS "Product class", P.PRODUCT_QUANTITY_AVAIL AS "Available Quantity",
CASE
  WHEN p.PRODUCT_QUANTITY_AVAIL < 20 THEN "LOW STOCK"
  WHEN (p.PRODUCT_QUANTITY_AVAIL > 21 AND p.PRODUCT_QUANTITY_AVAIL < 80) THEN "In Stock"
  WHEN p.PRODUCT_QUANTITY_AVAIL > 81 THEN "Enough Stock"
  WHEN p.PRODUCT_QUANTITY_AVAIL <= 0 THEN "Out of Stock"
END AS 'Inventory Status'
FROM product_class pc
JOIN product p ON pc.PRODUCT_CLASS_CODE = p.PRODUCT_CLASS_CODE
WHERE pc.PRODUCT_CLASS_DESC IN ('Clothes', 'Stationery')

UNION
```

```
#3c. Rest of the categories, if qty < 15 - 'Low Stock', 16 < qty < 50 - 'In Stock', > 51 - 'Enough stock'
SELECT p.PRODUCT_DESC AS "Product Name", pc.PRODUCT_CLASS_DESC AS 'Product Class', P.PRODUCT_QUANTITY_AVAIL AS 'Available Quantity',
CASE
WHEN p.PRODUCT_QUANTITY_AVAIL < 15 THEN "LOW STOCK"
WHEN (p.PRODUCT_QUANTITY_AVAIL > 16 AND p.PRODUCT_QUANTITY_AVAIL < 50) THEN "In STOCK"
WHEN p.PRODUCT_QUANTITY_AVAIL > 51 THEN "Enough Stock"
WHEN p.PRODUCT_QUANTITY_AVAIL <= 0 THEN "Out of Stock"
END AS 'Inventory Status'
FROM product_class pc
JOIN product p ON pc.PRODUCT_CLASS_CODE = p.PRODUCT_CLASS_CODE
WHERE pc.PRODUCT_CLASS_DESC NOT IN ('Computer', 'Electronics', 'Clothes', 'Stationery');
```

	Product Name	Product Class	Available Quantity	Inventory Status
▶	Sky LED 102 CM TV	Electronics	30	NULL
	Sams 192 L4 Single-door Refrigerator	Electronics	15	In STOCK
	Jocky Speaker Music System HT32	Electronics	19	In STOCK
	Logtech M244 Optical Mouse	Computer	10	NULL
	External Hard Disk 500 GB	Computer	10	NULL
	Cybershot DWC-W325 Camera	Electronics	5	LOW STOCK
	Huwi Wi-Fi Receiver 500Mbps	Computer	30	NULL
	Infant Sleepwear Blue	Clothes	50	In STOCK
	Blue Jeans 34	Clothes	100	Enough Stock

Fig 16: Query and output for determining the inventory status based on the availability of product quantity.

- Step21:** To generate a list of customers residing outside the state of 'Karnataka' who have not made any purchases of 'Toys' or 'Books'. The relevant columns were selected from the "online_customer" table, including customer information. The "state" column was selected from the "address" table. The "product_class_desc" column was selected from the "product_class". To establish the connection between "online_customer" and "address" tables an inner join was performed using the common key "address_id". As there was no direct linkage between the "product_class" and "online_customers" tables, additional tables such as "order_headers", "order_items", and "product" were utilized as intermediaries. The desired results were achieved by leveraging the intermediary tables to establish the

required associations. The WHERE clause was employed in conjunction with the “NOT IN” condition to filter out all states except ‘Karnataka’ and exclude products other than toys and books.

```
-- 4. List customers from outside Karnataka who haven't bought any toys or books
select oc.customer_id, oc.customer_fname, oc.customer_lname, oc.customer_email, oc.customer_phone, oc.customer_username, oc.customer_gender,
a.state, pc.PRODUCT_CLASS_DESC
from online_customer oc
join address a on oc.ADDRESS_ID = a.ADDRESS_ID
join order_header oh on oh.customer_id = oc.customer_id
join order_items oi on oh.order_id = oi.order_id
join product p on p.product_id = oi.product_id
join product_class pc on pc.product_class_code = p.product_class_code
where state not in ('Karnataka')
and PRODUCT_CLASS_DESC not in ('Toys','Books');
```

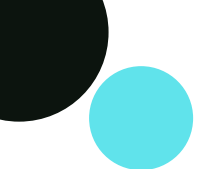
	customer_id	customer_fname	customer_lname	customer_email	customer_phone	customer_username	customer_gender	state	PRODUCT_CLASS_DESC
▶	3	Komal	Choudhary	ch_komal@yahoo.co.IN	9178234526	komalc	F	Delhi	Computer
	3	Komal	Choudhary	ch_komal@yahoo.co.IN	9178234526	komalc	F	Delhi	Stationery
	3	Komal	Choudhary	ch_komal@yahoo.co.IN	9178234526	komalc	F	Delhi	Stationery
	3	Komal	Choudhary	ch_komal@yahoo.co.IN	9178234526	komalc	F	Delhi	Stationery
	7	Ashwathi	Bhatt	ash_bhat@yahoo.co.IN	9773636307	abhata	F	Delhi	Kitchen Items
	7	Ashwathi	Bhatt	ash_bhat@yahoo.co.IN	9773636307	abhata	F	Delhi	Kitchen Items
	7	Ashwathi	Bhatt	ash_bhat@yahoo.co.IN	9773636307	abhata	F	Delhi	Stationery
	7	Ashwathi	Bhatt	ash_bhat@yahoo.co.IN	9773636307	abhata	F	Delhi	Stationery
	4	Wilfred	Jean	w_jean@gmail.com	9196257439	jeanw	M	Delhi	Electronics
	4	Wilfred	Jean	w_jean@gmail.com	9196257439	jeanw	M	Delhi	Electronics

Fig 17: Query and output to generate a list of customers who were residing outside of Karnataka and did not purchase books or toys.

What did you learn from solving it and how do you plan on using it in the future?

A. Retail Banking Case Study:

Through the "Retail Banking Case Study," we gained valuable insights into the criticality of missing data and its impact on extracting meaningful insights. A meticulous analysis of the data was emphasized to ensure that all the necessary information was present in the corresponding tables. The absence of values or common columns can introduce issues

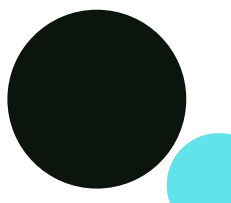


and potentially hinder the accuracy and reliability of the insights derived from the database.

Furthermore, we delved into the concept of "Triggers" and comprehended their significance within the context of the "Retail Banking Case Study." Triggers provided a mechanism to automatically update the values of a table as soon as relevant changes occurred in other associated tables. This understanding highlighted the essential role of triggers in maintaining data integrity and ensuring synchronization between entities within the database.

The case study effectively showcased how entities within the database were interconnected, enabling updates and changes in their values based on the values derived from different entities. This knowledge shed light on the complex relationships and dependencies among various elements of the database, facilitating a deeper understanding of the interplay between different entities and the impact on data consistency and accuracy.

Overall, the "Retail Banking Case Study" served as an instructive experience, emphasizing the significance of data completeness, the role of triggers in maintaining data consistency, and the interrelationships among entities within the database. It provided a comprehensive understanding of the challenges associated with missing data and the importance of establishing reliable mechanisms for data updates and synchronization.





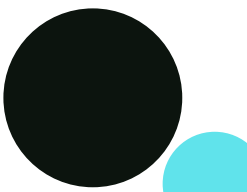
B. E-Commerce Retail Product Database Analysis:

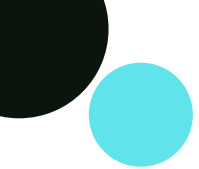
During the Product Database Analysis, we gained a comprehensive understanding of the significance of the MYSQL Entity Relationship Diagram (ERD). In the given case study, the default lines representing the relationships between different entities were absent. As a result, we manually drew the lines to establish the relationships and create a visual representation of the connections between tables.

By manually creating the relationship lines, we were able to enhance the readability and relevance of the ERD. This enabled us to easily discern the connections between the various tables and entities in a more intuitive manner. The ERD played a crucial role in facilitating our comprehension of the data structure and its interdependencies.

The case study served as a valuable lesson on how an ER diagram can greatly impact the analysis and understanding of data. It emphasized the importance of having a well-defined and accurate ERD to guide our exploration and interpretation of the database. The ERD served as a powerful tool for identifying common linking points between two entities, which in turn significantly influenced the extraction of meaningful insights from the data.

Thus, from both the case studies we learned different things and, we understood what the pre-requisites are that we need





to ensure before beginning the analyzation and extraction of meaningful insights/results from the database.

What could have been done better?

Missing values in the dataset:

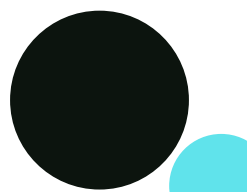
- The given data set had missing values for the city attribute in the customers table. We had to obtain the missing values by assuming that the city was Delhi and adding those values to get the output. We feel that having a complete and accurate data set is crucial for getting insights that are reliable.

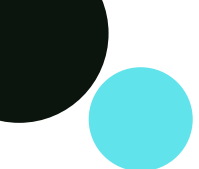
The tables in the database lacked proper interconnections:

- The given data set had missing values for the city attribute in the customers table. We had to obtain the missing values by assuming that the city was Delhi and adding those values to get the output. We feel that having a complete and accurate data set is crucial for getting insights that are reliable.

Example:

Retail banking case study:

- There was no direct linkage between the customers and branches table. We had to leverage the accounts table
- 



as an intermediary to establish the necessary connection.

E-Commerce Retail Product Database:

- The order_header table lacked direct_links to the 'product' or 'product_class' table. So, we had to use the "order_items" as an intermediary.
 - There was no direct linkage between the "product_class" and "online_customers" tables, additional tables such as "order_headers", "order_items", and "product" were utilized as intermediaries to achieve desired results.
 - Having interlinked tables in a database is important because it simplifies data management and improves efficiency. When tables are connected through common references, we can avoid complex queries and lengthy manual work.
- 