​

Personal Money Budget App



Team uncountables

**Team Members:**

Sourabh wadhwa -

Ananya Kondiparthy -

Mytreyi Reddy - W1573480

**INDEX**

**Final Report Contents:**

* Business application description
* Swim Lane Diagrams
* Business process of the application
* Describe business metrics that will be generated
* Logical schema – Data model
* Physical schema – Database dictionary
* DML in form of a SQL script
* Database backup file with sample business data
* SQL queries for each table – create, insert, update, and delete
* SQL queries for business metrics
* Include results of running the query
* Include business metrics reports
* For delete & update queries show rows in table before and after delete query
* Project summary

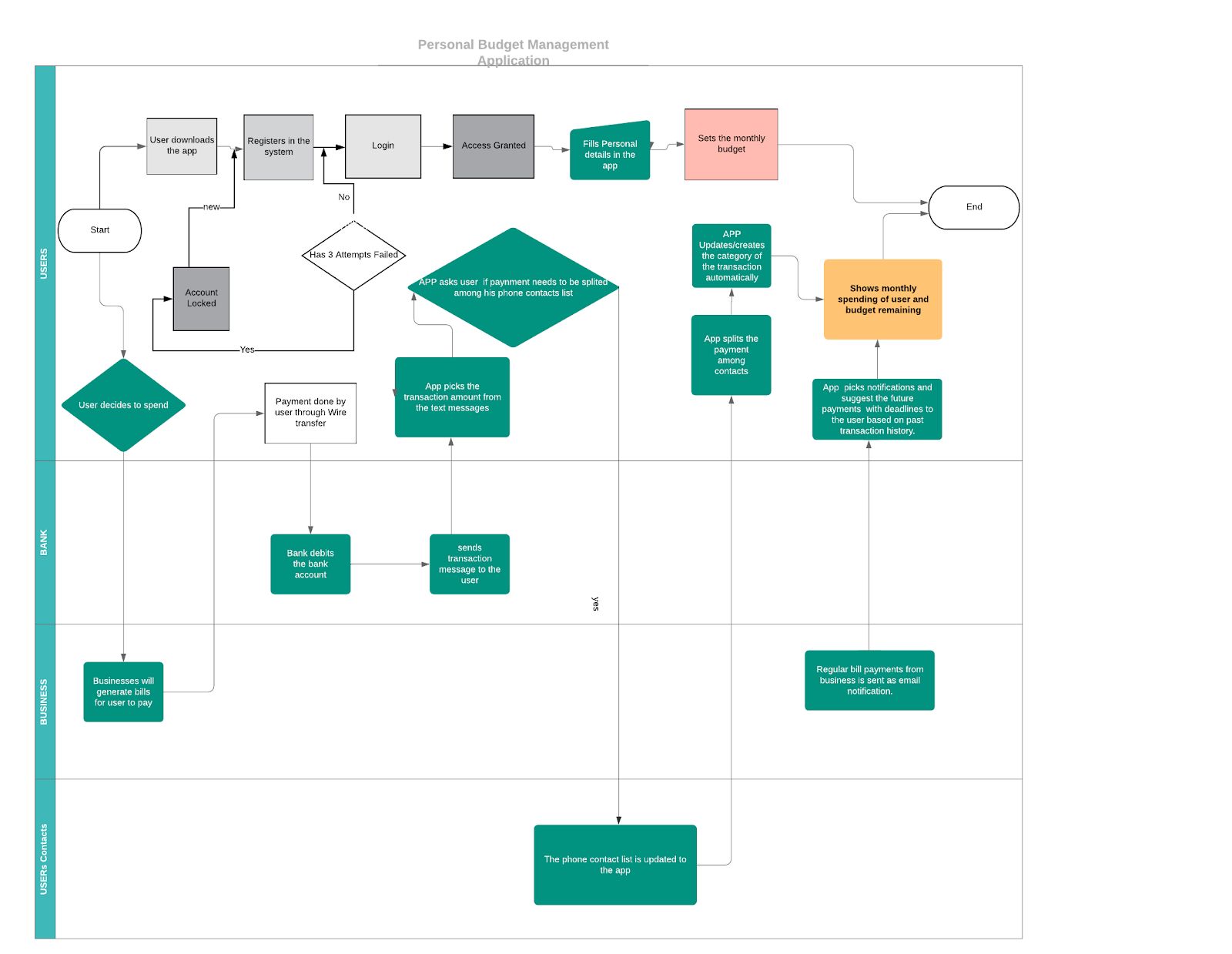
▪Summarize your experience with this exercise  
▪What was the hardest part of this project?  
▪What problems did you run against in this project?  
▪How did you solve these problems?  
▪If you were to do this project again, what methodology would you follow ▪Suggestions for how to refine this project for the next class

**1. Business application description:**

Budget Management Application Domain​: Financial management service

The project is about the database for a mobile app that will help its users in personal financial budget management. The users will be asked to enter personal information like name, age, financial goals, etc. Also, we will have options to store user's bank details, credit card details, and the loan details so any transactions from any of these can be fed into the app automatically. All this information can be saved at the time of registration (Sign up) to the app and also any other time user wants to update it. The expenses that are made using the bank accounts and cards of the user that are linked with this app will automatically be updated as "Transactions". These transactions are classified in "categories of purchases" (for example, rent, entertainment, utilities, travel expenses, education, etc). All these modes can be saved in the app for monitoring the expenses that a user makes. In addition, the user can save the recurring expenses like example monthly mobile bill payments, credit card dues, and loan installments. Using this data app will give a reminder to users to pay before the due date which we term as "Future Payments". Overall, this business application will give a clear view of the user’s budget and expenses. This can give us many interesting analytical findings like most spent categories, week wise spending’s, etc. Also, we can club many such reports to give the user an interactive dashboard of the transaction for a current and recent month.

**2. Swim Lane Diagrams:**

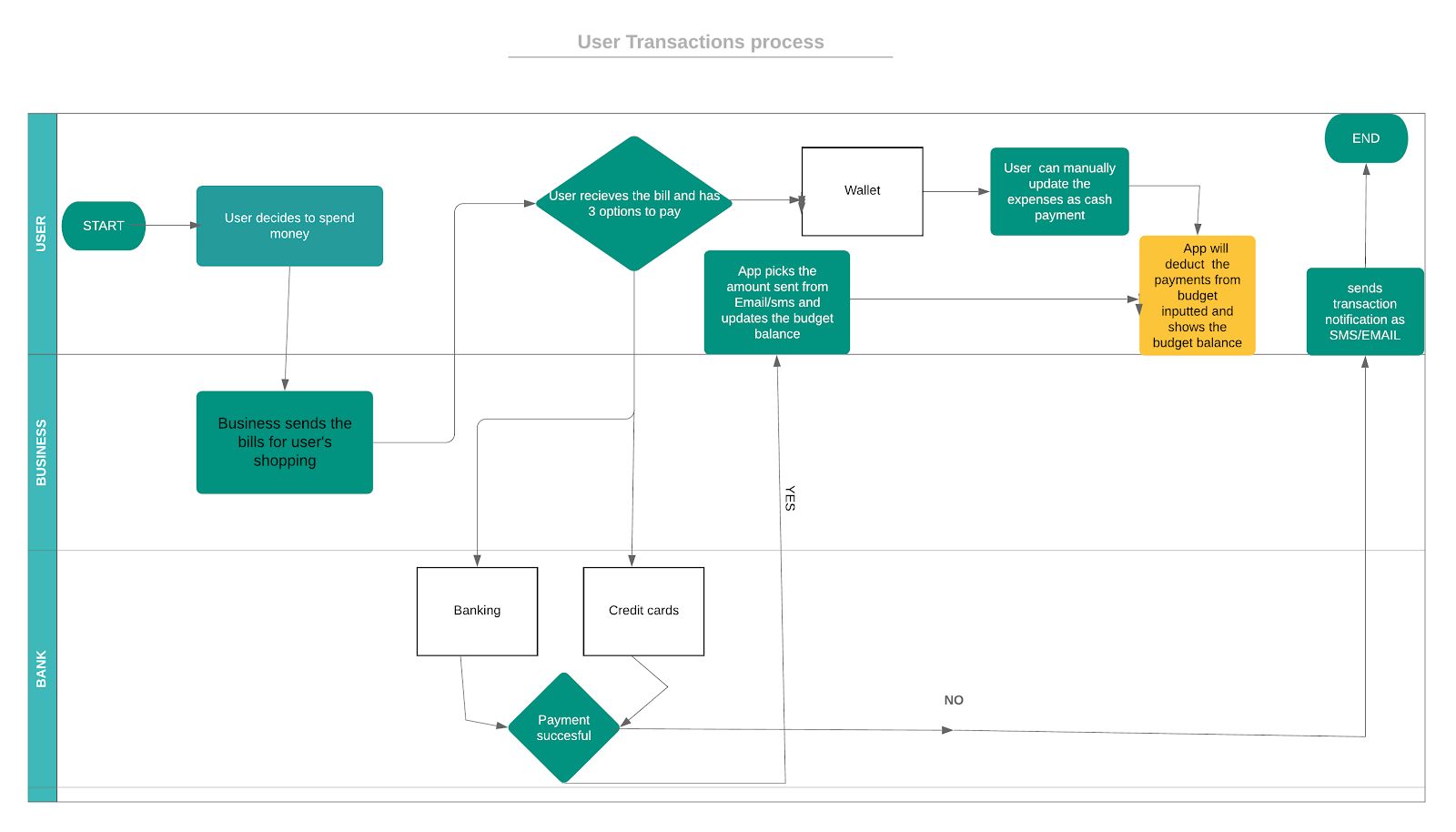


**Use cases:**

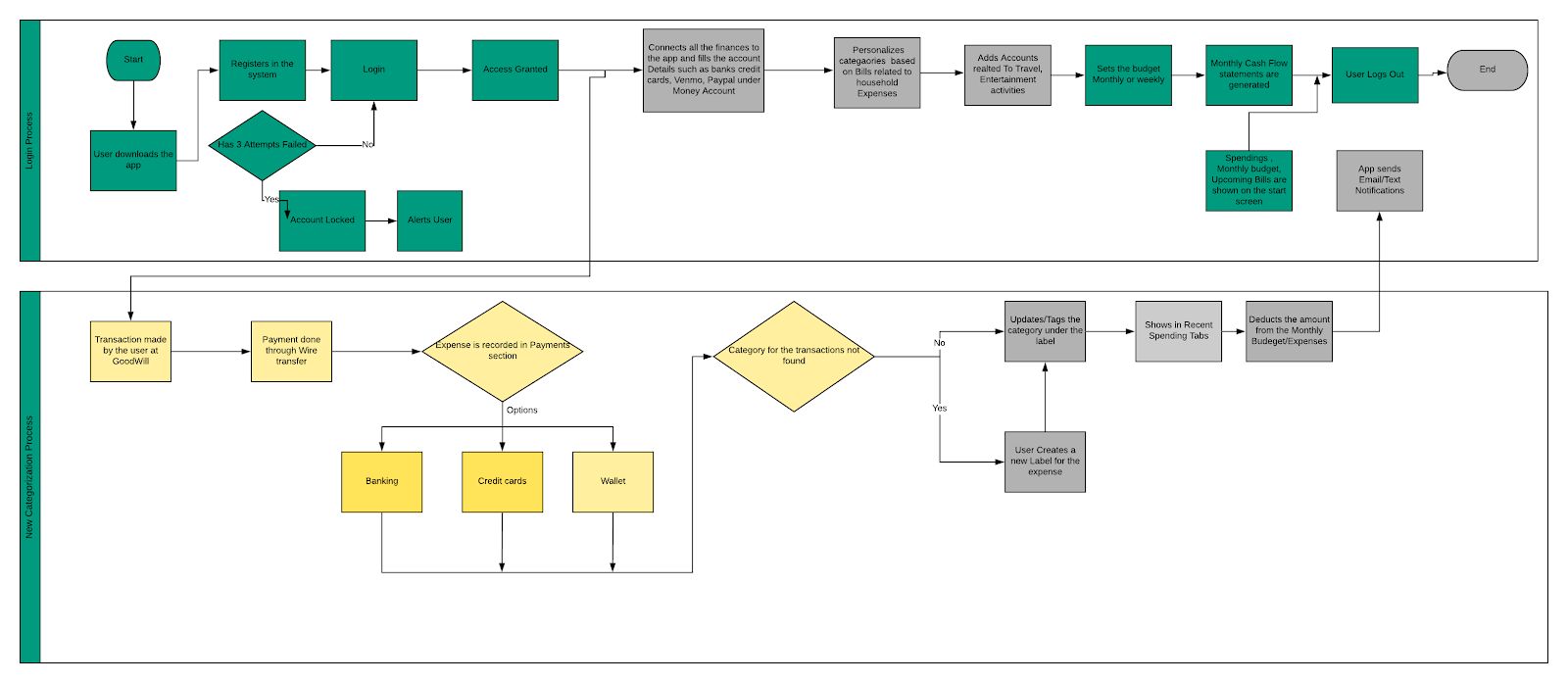
**Case 1: Login Process:**

User 1 downloads the app , subscribe to the service and then enters his personal details and financial information. User can maintain a passbook which can have multiple payments such as Wallet, net banking, credit cards so that he can track his accounts or balances credits and debits.

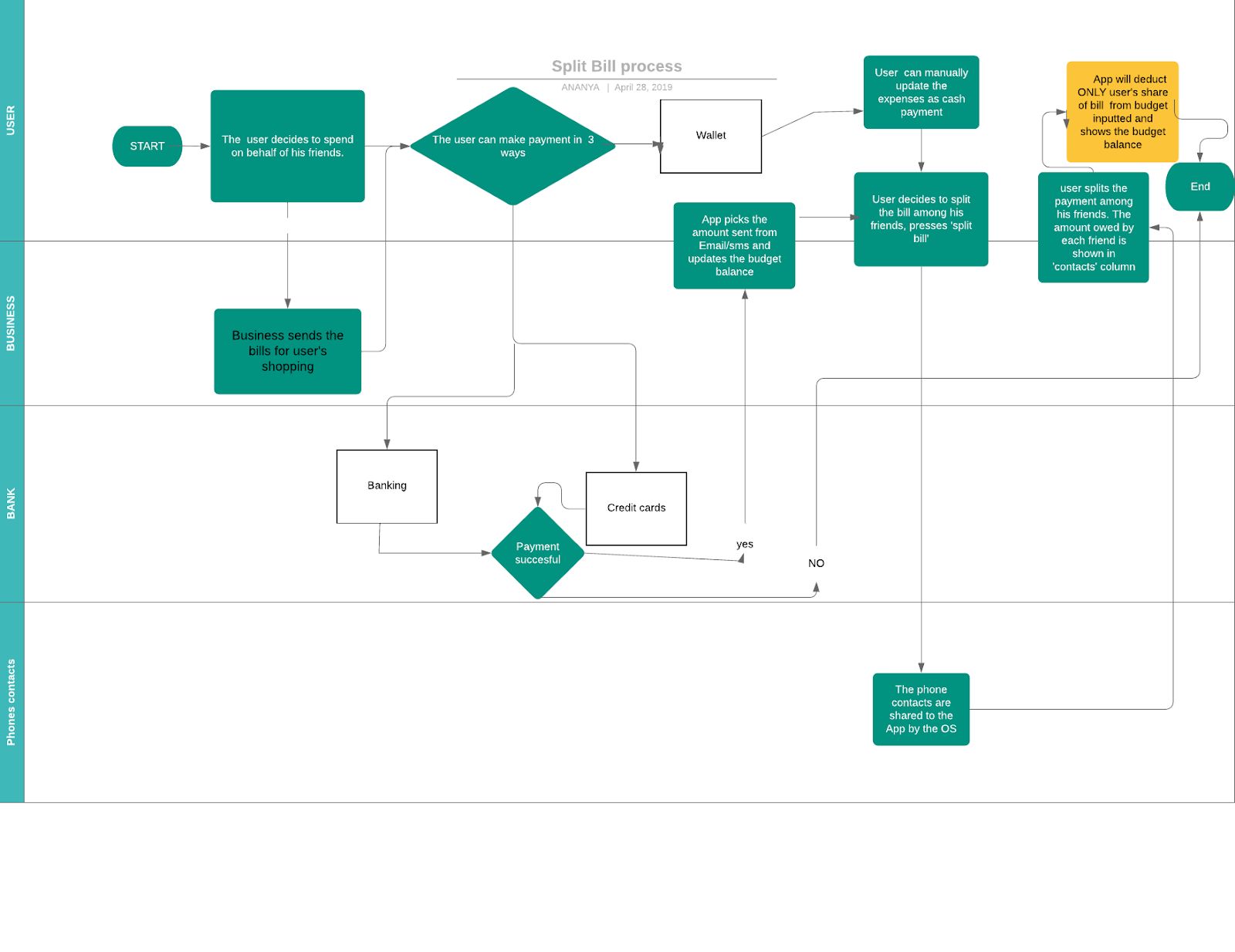
**Case 2:**​ ​**User Transaction Process:**John goes to Target to purchase grocery for his home. He buys milk, bread which totalled to amount $20. John pays the bill with his credit card and immediately receives a SMS and Email from his credit card company. The app recognises from the transaction SMS/ Email and automatically categorises as ‘Grocery Expense’. The App reduces the grocery budget set by john by $20.



**Case 3:**​ ​**New Categorization Process:**User\_ 3 made a donation at GoodWill. The transaction is done through bank wire transfer and the details related to the payment is not specifically categorized in the system. In such a scenario, user can create his own category of spending area, so that transactions related to donations or any service related payments will fall into the specific section.



**Case 4** ​: ​**User Split Bill Process:**John and his three friends go out lunch where their total bill was $80 which he pays using his credit card. This bill should be equally divided among all of them. They all decided to pay him later. The app recognizes from the transactions sms received on John’s mobile phone and marks it as expense. John updates the transaction to mark it as split among contacts and selects his friends from his contact list. App marked that amount as loan given for these contacts.



**Analytics use case:**

1. Create a report to demonstrate the category wise spendings for the user
2. Create a report to display the summary of previous months spendings.
3. Create a chart to display the week wise spendings.
4. Report demonstrating the most spent categories and its comparison with previous months.
5. Create a chart to display the months in which user overrun the budget.

**SQL queries for each table – create, insert, update, and delete:**

CREATE TABLE `user\_details` (

`User\_id` int(11) NOT NULL,

`First\_Name` varchar(45) DEFAULT NULL,

`Last\_name` varchar(45) DEFAULT NULL,

`Phone` varchar(15) DEFAULT NULL,

`Email` varchar(45) DEFAULT NULL,

`Address` varchar(100) DEFAULT NULL,

`Created\_Date` datetime DEFAULT NULL,

`Last\_Updated\_Date` datetime DEFAULT NULL,

`monthly\_budget` decimal(10,3) DEFAULT NULL,

PRIMARY KEY (`User\_id`)

)

CREATE TABLE `user\_login\_details` (

`User\_id` int(11) NOT NULL,

`Login\_id` varchar(45) NOT NULL,

`Password` varchar(45) DEFAULT NULL,

`Security\_Question\_1` int(11) DEFAULT NULL,

`Answer\_1` varchar(45) DEFAULT NULL,

`Security\_Question\_2` int(11) DEFAULT NULL,

`Answer\_2` varchar(45) DEFAULT NULL,

`Password\_Change\_Date` date DEFAULT NULL,

`Created\_Date` date DEFAULT NULL,

`Last\_Updated\_Date` date DEFAULT NULL,

PRIMARY KEY (`User\_id`,`Login\_id`),

CONSTRAINT `R\_6` FOREIGN KEY (`User\_id`) REFERENCES `user\_details` (`User\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `user\_categories` (

`User\_id` int(11) NOT NULL,

`User\_Category\_ID` int(11) NOT NULL,

`Category\_Desc` varchar(50) DEFAULT NULL,

`Created\_Date` date DEFAULT NULL,

`Last\_Updated\_Date` date DEFAULT NULL,

PRIMARY KEY (`User\_Category\_ID`),

KEY `R\_8` (`User\_id`),

CONSTRAINT `R\_8` FOREIGN KEY (`User\_id`) REFERENCES `user\_details` (`User\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `transaction\_split\_details` (

`Transaction\_ID` int(11) NOT NULL,

`Split\_ID` int(11) NOT NULL,

`Split\_with\_contact` varchar(50) DEFAULT NULL,

`Split\_amount` decimal(10,2) DEFAULT NULL,

`Created\_Date` date DEFAULT NULL,

`Last\_Updated\_Date` date DEFAULT NULL,

PRIMARY KEY (`Transaction\_ID`,`Split\_ID`),

CONSTRAINT `R\_10` FOREIGN KEY (`Transaction\_ID`) REFERENCES `transaction\_details` (`Transaction\_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `transaction\_details` (

`Payment\_method\_id` int(11) NOT NULL,

`Transaction\_ID` int(11) NOT NULL,

`Category\_ID` int(11) DEFAULT NULL,

`Transaction\_Date` date DEFAULT NULL,

`Transaction\_Amt` decimal(10,3) DEFAULT NULL,

`Created\_Date` date DEFAULT NULL,

`Last\_Updated\_date` date DEFAULT NULL,

PRIMARY KEY (`Transaction\_ID`),

KEY `R\_9` (`Payment\_method\_id`),

CONSTRAINT `R\_9` FOREIGN KEY (`Payment\_method\_id`) REFERENCES `payment\_method\_details` (`Payment\_method\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `security\_questions` (

`Question\_ID` int(11) NOT NULL,

`Security\_Q\_Desc` varchar(100) DEFAULT NULL,

`Created\_Date` date DEFAULT NULL,

`Last\_Updated\_Date` date DEFAULT NULL,

PRIMARY KEY (`Question\_ID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `payment\_method\_details` (

`User\_id` int(11) NOT NULL,

`Payment\_method\_id` int(11) NOT NULL,

`Pymt\_method\_type` varchar(1) DEFAULT NULL,

`Created\_Date` date DEFAULT NULL,

`Last\_Updated\_Date` date DEFAULT NULL,

PRIMARY KEY (`Payment\_method\_id`),

KEY `R\_7` (`User\_id`),

CONSTRAINT `R\_7` FOREIGN KEY (`User\_id`) REFERENCES `user\_details` (`User\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `future\_bills` (

`User\_id` int(11) DEFAULT NULL,

`Bill\_id` int(11) NOT NULL,

`Invoice\_number` varchar(25) DEFAULT NULL,

`Bill\_date` date DEFAULT NULL,

`Due\_date` date DEFAULT NULL,

`Bill\_amount` decimal(10,3) DEFAULT NULL,

`Created\_date` date DEFAULT NULL,

`Last\_update\_date` date DEFAULT NULL,

PRIMARY KEY (`Bill\_id`),

KEY `R\_23` (`User\_id`),

CONSTRAINT `R\_23` FOREIGN KEY (`User\_id`) REFERENCES `user\_details` (`User\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `credit\_card\_details` (

`Payment\_method\_id` int(11) NOT NULL,

`Card\_id` int(11) DEFAULT NULL,

`Credit\_card\_number` varchar(20) DEFAULT NULL,

`Credit\_card\_expiry\_date` date DEFAULT NULL,

`Issuing\_Company` varchar(100) DEFAULT NULL,

`Created\_date` date DEFAULT NULL,

`Last\_Updated\_Date` date DEFAULT NULL,

PRIMARY KEY (`Payment\_method\_id`),

CONSTRAINT `R\_22` FOREIGN KEY (`Payment\_method\_id`) REFERENCES `payment\_method\_details` (`Payment\_method\_id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `categories` (

`Category\_id` int(11) NOT NULL,

`Category\_Desc` varchar(50) DEFAULT NULL,

`Created\_Date` date DEFAULT NULL,

`Last\_Updated\_Date` date DEFAULT NULL,

PRIMARY KEY (`Category\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `bank\_details` (

`Payment\_method\_id` int(11) NOT NULL,

`Bank\_id` int(11) DEFAULT NULL,

`Bank\_Name` varchar(50) DEFAULT NULL,

`Account\_Number` varchar(50) DEFAULT NULL,

`Created\_Date` date DEFAULT NULL,

`Last\_Updated\_Date` date DEFAULT NULL,

PRIMARY KEY (`Payment\_method\_id`),

CONSTRAINT `R\_15` FOREIGN KEY (`Payment\_method\_id`) REFERENCES `payment\_method\_details` (`Payment\_method\_id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `audit\_transactions` (

`Payment\_method\_id` int(11) NOT NULL,

`Transaction\_ID` int(11) NOT NULL,

`New\_Transaction\_ID` int(11) NOT NULL AUTO\_INCREMENT,

`Category\_ID` int(11) DEFAULT NULL,

`Transaction\_Date` date DEFAULT NULL,

`Transaction\_Amt` decimal(10,3) DEFAULT NULL,

`New\_Transaction\_Amt` decimal(10,3) DEFAULT NULL,

`Record\_Status` varchar(10) DEFAULT NULL,

`First\_Created\_Date` date DEFAULT NULL,

`updated\_at` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`updated\_by` varchar(30) NOT NULL,

PRIMARY KEY (`New\_Transaction\_ID`),

KEY `Payment\_method\_id` (`Payment\_method\_id`),

CONSTRAINT `audit\_transactions\_ibfk\_1` FOREIGN KEY (`Payment\_method\_id`) REFERENCES `payment\_method\_details` (`Payment\_method\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

select \* from user\_details;

A screenshot of a social media post

Description automatically generated

select \* from user\_login\_details;

A screenshot of a social media post

Description automatically generated

select \* from user\_categories;

A screenshot of a social media post

Description automatically generated

select \* from transaction\_split\_details;

A screenshot of a computer

Description automatically generated

select \* from transaction\_details;

A screenshot of a computer

Description automatically generated

select \* from security\_questions;

A screenshot of a social media post

Description automatically generated

select \* from payment\_method\_details;

A screenshot of a computer

Description automatically generated

select \* from future\_bills;

A screenshot of a computer

Description automatically generated

select \* from credit\_card\_details;

A screenshot of a computer

Description automatically generated

select \* from categories;

A screenshot of a social media post

Description automatically generated

select \* from bank\_details;

A screenshot of a computer

Description automatically generated

select \* from audit\_transactions;

A screenshot of a computer

Description automatically generated

**Function max\_spent\_category:**

DELIMITER ;;

CREATE FUNCTION `max\_spent\_category`(userid int)

RETURNS int(11)

BEGIN

DECLARE category int;

DECLARE max\_amt decimal(10,3);

select cat\_wise\_sum.category\_id into category from

(select ud.user\_Id,td.category\_id,td.Month\_Year,sum(td.transaction\_amt) total\_transations from

user\_details ud, Payment\_method\_details pmd,

(select concat(MONTHNAME(Transaction\_Date), '-', year(Transaction\_Date)) as' Month\_Year',td.\* from Transaction\_details td) td

where ud.user\_id = pmd.user\_id

and pmd.payment\_method\_id = td.payment\_method\_id

group by ud.user\_Id,td.category\_id,td.Month\_Year) cat\_wise\_sum,

(select ud.user\_id , max(td.transaction\_amt) max\_transations from

user\_details ud, Payment\_method\_details pmd,

(select concat(MONTHNAME(Transaction\_Date), '-', year(Transaction\_Date)) as' Month\_Year',DATE\_FORMAT(Transaction\_Date,'%Y%m') as 'yearmonth',td.\* from Transaction\_details td) td,

(select ud.user\_id, MAX(DATE\_FORMAT(Transaction\_Date,'%Y%m')) LATEST\_MONTH from transaction\_details tran, payment\_method\_details pmd, user\_details ud

GROUP BY ud.user\_id) LMU

where ud.user\_id = pmd.user\_id

and pmd.payment\_method\_id = td.payment\_method\_id

and LMU.LATEST\_MONTH = td.yearmonth

and LMU.user\_id = ud.user\_id

and ud.user\_id = userid

group by ud.user\_Id) max\_spent

where cat\_wise\_sum.user\_id = max\_spent.user\_id

and cat\_wise\_sum.total\_transations = max\_spent.max\_transations;

RETURN (category);

END ;;

DELIMITER ;

**Stored Procedure check\_budget\_overrun:**

DELIMITER ;;

CREATE PROCEDURE `check\_budget\_overrun`( IN Userid int(10), IN present\_month date, OUT Budget\_over varchar(10) )

begin

select

case

when monthly\_budget <= (total\_transations + loan\_amt) then 'In budget'

else 'Over Budget'

end into Budget\_over

from (

select ud.user\_Id,td.Month\_Year, ud.monthly\_budget, sum(td.transaction\_amt) total\_transations,sum(sd.loan\_amt) loan\_amt from

user\_details ud, Payment\_method\_details pmd,

(select concat(MONTHNAME(Transaction\_Date), '-', year(Transaction\_Date)) as' Month\_Year',td.\* from Transaction\_details td) td,

(select sum(Split\_amount) loan\_amt,Transaction\_ID from transaction\_split\_details group by Transaction\_ID) sd

where ud.user\_id = pmd.user\_id

and pmd.payment\_method\_id = td.payment\_method\_id

and ud.User\_id = Userid

and td.Month\_Year = concat(MONTHNAME(present\_month), '-', year(present\_month))

and td.Transaction\_ID = sd.Transaction\_ID

group by ud.user\_Id,td.Month\_Year) monthly\_data;

end ;;

DELIMITER ;

**Update Trigger Audit\_Transaction\_Detail**

DELIMITER $$

create trigger Audit\_Transaction\_Detail

before update on transaction\_details

for each row begin insert into Audit\_Transactions

( Payment\_method\_id, Transaction\_ID,Category\_ID, Transaction\_Date,

Transaction\_Amt, New\_Transaction\_Amt, Record\_Status, First\_Created\_Date,updated\_by

)

values

(old.Payment\_method\_id, old.Transaction\_ID,

old.Category\_ID, old.Transaction\_Date,

old.Transaction\_Amt, new.Transaction\_Amt, "Update", old.Created\_Date,

user());

END$$

DELIMITER ;

**Delete Trigger**

DELIMITER $$

CREATE TRIGGER delete\_transaction

Before DELETE ON transaction\_details

for each row

begin

delete from transaction\_split\_details where Transaction\_ID = old.Transaction\_ID;

insert into Audit\_Transactions

( Payment\_method\_id, Transaction\_ID,Category\_ID, Transaction\_Date,

Transaction\_Amt,Record\_Status, First\_Created\_Date,updated\_by

)

values

(old.Payment\_method\_id, old.Transaction\_ID,

old.Category\_ID, old.Transaction\_Date,

old.Transaction\_Amt, "Delete", old.Created\_Date,

user());

END$$

DELIMITER ;

**View `error\_transaction`**

create VIEW `error\_transaction` AS select `td`.`Transaction\_ID` AS `transaction\_id`,

`tsd`.`Split\_ID` AS `split\_id`,

`td`.`Transaction\_Amt` AS `Transaction\_Amt`,

`tsd`.`Split\_amount` AS `Split\_amount`

from (`transaction\_details` `td` join `transaction\_split\_details` `tsd`)

where ((`td`.`Transaction\_ID` = `tsd`.`Transaction\_ID`) and (`td`.`Transaction\_Amt` < `tsd`.`Split\_amount`));

**VIEW `multi\_transaction`**

create VIEW `multi\_transaction` AS select count(0) AS `count(\*)`,

`transaction\_split\_details`.`Transaction\_ID` AS `transaction\_id`

from `transaction\_split\_details`

group by `transaction\_split\_details`.`Transaction\_ID`

having (count(0) > 1);

/\* Analytics use case:

**1. Create a report to demonstrate the category wise spendings for the user** \*/

----------------Case 1 ---------------------

select Category\_Desc, sum(Transaction\_Amt) as Total\_Spendings, pmd.User\_id

from categories cat, transaction\_details tran, payment\_method\_details pmd, user\_details ud

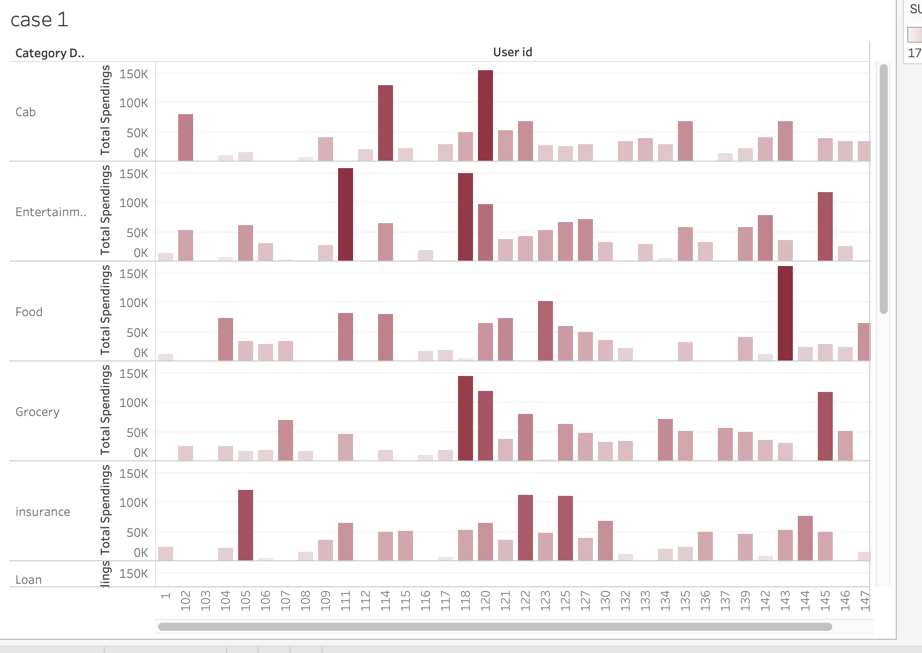
where (cat.Category\_id = tran.Category\_ID) and (pmd.Payment\_method\_id = tran.Payment\_method\_id)

group by cat.Category\_Desc, pmd.User\_id

order by Category\_Desc;

A screenshot of a social media post

Description automatically generated



2. Create a report to display the summary of previous months spendings.

select ud.user\_Id,td.Month\_Year,sum(td.transaction\_amt) total\_transations from

user\_details ud, Payment\_method\_details pmd,(select concat(MONTHNAME(Transaction\_Date), '-', year(Transaction\_Date)) as' Month\_Year',td.\* from Transaction\_details td) td

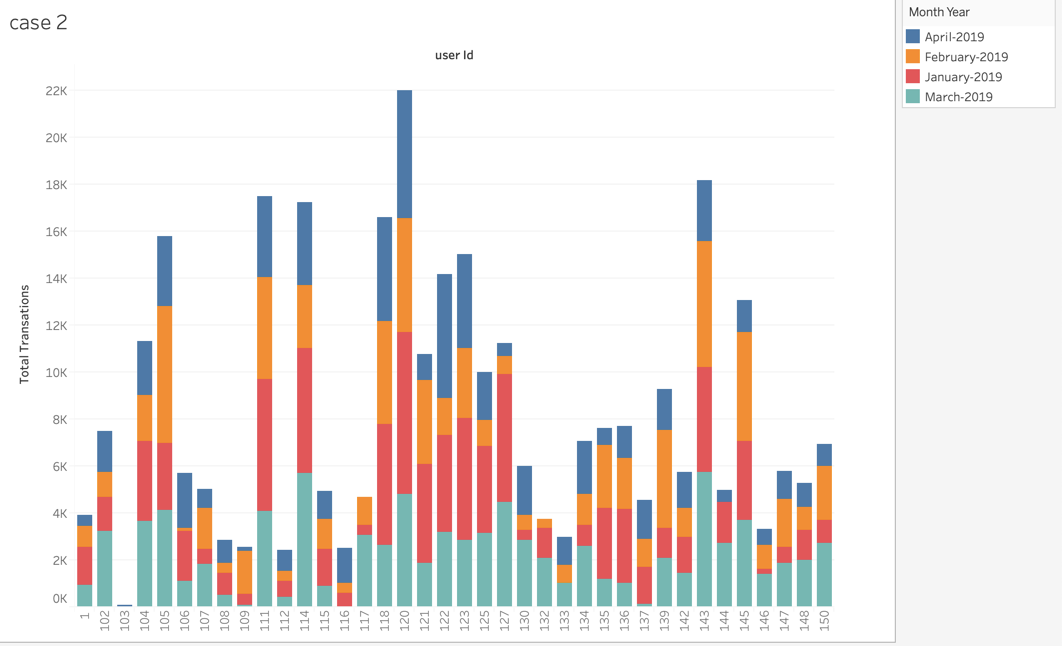
where ud.user\_id = pmd.user\_id

and pmd.payment\_method\_id = td.payment\_method\_id

group by ud.user\_Id,td.Month\_Year;

A screenshot of a social media post

Description automatically generated



3. Create a chart to display the week wise spendings.

select ud.user\_Id,td.trans\_week,sum(td.transaction\_amt) total\_transations from

user\_details ud, Payment\_method\_details pmd,

(select week(Transaction\_Date) as 'trans\_week',td.\* from Transaction\_details td) td

where ud.user\_id = pmd.user\_id

and pmd.payment\_method\_id = td.payment\_method\_id

group by ud.user\_Id,td.trans\_week

order by ud.user\_Id asc,trans\_week desc;

A screenshot of a social media post

Description automatically generated

4. Report demonstrating the most spent categories and its comparison with previous months.

select ud.user\_Id,(select Category\_Desc from categories c where c.Category\_id = td.Category\_id) as category,td.Month\_Year,sum(td.transaction\_amt) total\_transations from

user\_details ud, Payment\_method\_details pmd,

(select concat(MONTHNAME(Transaction\_Date), '-', year(Transaction\_Date)) as' Month\_Year',DATE\_FORMAT(Transaction\_Date,'%Y%m') as 'yearmonth',td.\* from Transaction\_details td) td

where ud.user\_id = pmd.user\_id

and pmd.payment\_method\_id = td.payment\_method\_id

and (td.category\_id, ud.user\_id ) in

(select cat\_wise\_sum.category\_id ,cat\_wise\_sum.user\_id from

(select ud.user\_Id,td.category\_id,td.Month\_Year,sum(td.transaction\_amt) total\_transations from

user\_details ud, Payment\_method\_details pmd,

(select concat(MONTHNAME(Transaction\_Date), '-', year(Transaction\_Date)) as' Month\_Year',td.\* from Transaction\_details td) td

where ud.user\_id = pmd.user\_id

and pmd.payment\_method\_id = td.payment\_method\_id

group by ud.user\_Id,td.category\_id,td.Month\_Year) cat\_wise\_sum,

(select ud.user\_id , max(td.transaction\_amt) max\_transations from

user\_details ud, Payment\_method\_details pmd,

(select concat(MONTHNAME(Transaction\_Date), '-', year(Transaction\_Date)) as' Month\_Year',DATE\_FORMAT(Transaction\_Date,'%Y%m') as 'yearmonth',td.\* from Transaction\_details td) td,

(select ud.user\_id, MAX(DATE\_FORMAT(Transaction\_Date,'%Y%m')) LATEST\_MONTH from transaction\_details tran, payment\_method\_details pmd, user\_details ud

GROUP BY ud.user\_id) LMU

where ud.user\_id = pmd.user\_id

and pmd.payment\_method\_id = td.payment\_method\_id

and LMU.LATEST\_MONTH = td.yearmonth

and LMU.user\_id = ud.user\_id

group by ud.user\_Id) max\_spent

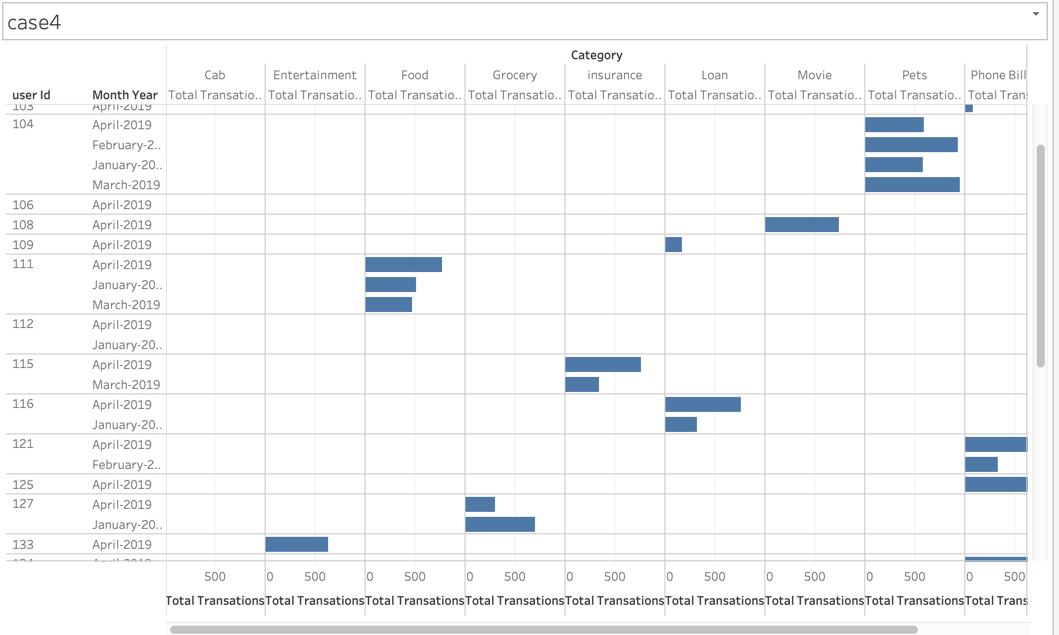
where cat\_wise\_sum.user\_id = max\_spent.user\_id

and cat\_wise\_sum.total\_transations = max\_spent.max\_transations)

group by ud.user\_Id,td.Month\_Year

order by ud.user\_id asc,td.yearmonth desc

;



5. Create a chart to display the months in which user overrun the budget.

select user\_id,Month\_Year,

case

when monthly\_budget <= (total\_transations + loan\_amt) then 'In budget'

else 'Over Budget'

end as budget\_overflow

from (

select ud.user\_Id,td.Month\_Year, ud.monthly\_budget, sum(td.transaction\_amt) total\_transations,sum(sd.loan\_amt) loan\_amt from

user\_details ud, Payment\_method\_details pmd,

(select concat(MONTHNAME(Transaction\_Date), '-', year(Transaction\_Date)) as' Month\_Year',td.\* from Transaction\_details td) td,

(select sum(Split\_amount) loan\_amt,Transaction\_ID from transaction\_split\_details group by Transaction\_ID) sd

where ud.user\_id = pmd.user\_id

and pmd.payment\_method\_id = td.payment\_method\_id

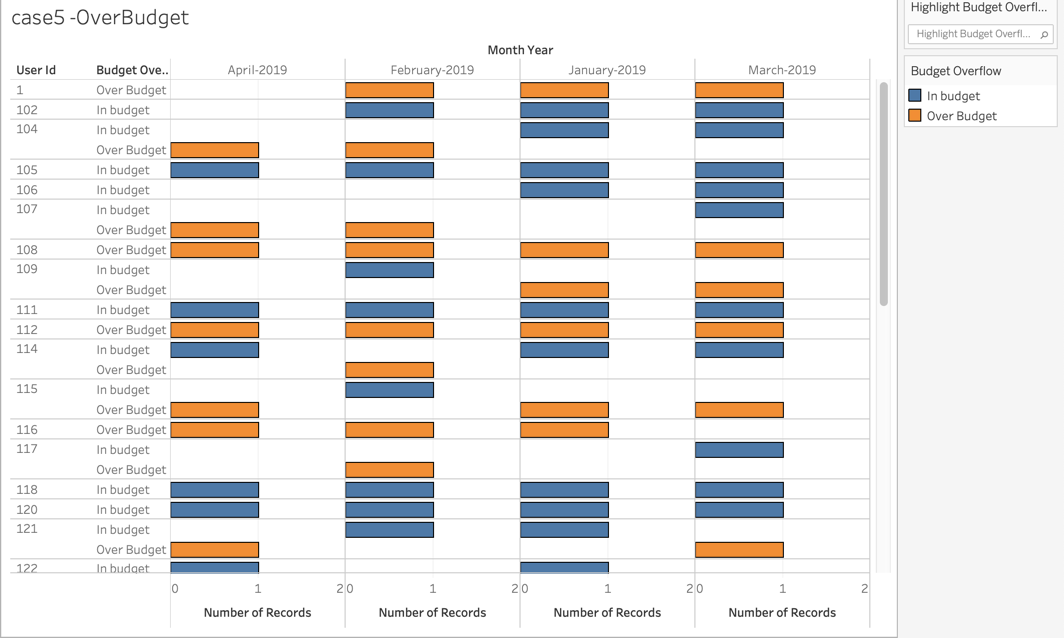
and td.Transaction\_ID = sd.Transaction\_ID

group by ud.user\_Id,td.Month\_Year) monthly\_data;

A screenshot of a social media post

Description automatically generated

Tableau Report:



Project summary :

* Understanding the business is equally important to implementing the technical model.
* Problems: 1. Had to change the data model as we have missed few of the attributes while creating the entities. 2. Inconsistency data problems while uploading the data. Example Split amount was more the original transaction amount.
* To solve the first problem, we sat together and re-iterated to the functionality and changed the model according to the discussion and moved forward.
* To solve the second problem, we wrote a query to find out all those records and updated them in a way that they will have a portion of transaction amount.
* If we were to do the project again the we will spend more time in the initial phase of the project and make sure to double check the output of each phase.