* To generate unique Ids in AppSheet

CONCATENATE("SC", RIGHT("000" & (COUNT(EMPLOYEES[Empid]) + 1), 3))

* Validate the uniqueness of Employee IDs using the Valid\_If constraint in the column definition

ISBLANK(SELECT(EMPLOYEES[Empno], [Empno] = [\_THISROW].[Empno]))

* TO Extract details using IDs or any Reference

LOOKUP([\_THISROW].[Empno],"Employees","Empno","Employee Name")

* To Get Data/any Details from another Table based on condition

any(select(Check Out Time[Out Time],and([\_THISROW].[Empno]=[Empno], [\_THISROW].[Attendance Date]=[Attendance Date])))

any(select(Check In Time[Check-In-DateTime],and([\_THISROW].[Empno]=[Empno], [\_THISROW].[ID]=[ID])))

any(select(Check Out Time[Check-Out-DT], and([\_THISROW].[Empno] = [Empno], [\_THISROW].[ID]=[ID])))

* To avoid duplicate entries

ISBLANK(select(Check In Time[Empno],and([Empno]=[\_THISROW].[Empno],[Attendance Date]=[\_THISROW].[Attendance Date])))  
  
ISBLANK(

FILTER(

"Check In Time",

AND(

[Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = DATE([\_THISROW].[Check-In-DateTime])

)

)

)

* **TO GET DUPLICATE ENTRIES MULTIPLE TIMES BUT ON A CONDITION**

ISBLANK(FILTER("Check In Time",AND([Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = DATE([\_THISROW].[Check-In-DateTime]),ISBLANK([Out-DateTime]))))

[ **CAN CHECK IN AGAIN ONLY AFTER HE MADE CHECK OUT OF HIS PREVIOUS CHECK IN** ]

* Combining these conditions ensures that an employee can only check in if they have checked out on the same day and have also checked out from the previous day. This prevents scenarios where an employee checks in on a new day without having properly checked out from the previous day.

AND(ISBLANK(FILTER("Check In Time",AND([Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = DATE([\_THISROW].[Check-In-DateTime]),

ISBLANK([Out-DateTime]) ))),

ISBLANK(FILTER("Check In Time",AND( [Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = (DATE([\_THISROW].[Check-In-DateTime]) - 1),

ISBLANK([Out-DateTime])))))

SECOND FORMULA TO ENSURE THAT AN EMPLOYEE CANNOT CHECK IN IF THEY HAVE ANY OPEN CHECK-INS REGARDLESS OF THE DATE.

AND(ISBLANK(FILTER("Check In Time",AND([Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = DATE([\_THISROW].[Check-In-DateTime]),

ISBLANK([Out-DateTime]) ))),

ISBLANK(FILTER("Check In Time",AND( [Empno] = [\_THISROW].[Empno],

ISBLANK([Out-DateTime])))))

With this formula, any attempt by an employee to check in will be blocked if they have any previous check-ins that do not have a corresponding check-out time. This ensures strict enforcement of the rule that each check-in must be followed by a check-out before a new check-in can occur, regardless of the number of days passed.

### **Example**

* If an employee checks in on May 1 and does not check out, they will not be allowed to check in again on May 2, May 3, or any subsequent day until they complete the check-out for the May 1 check-in.
* This formula calculates the hours worked only when the "Out Time" is entered. If "Out Time" is empty, it will display nothing in the "Calculated Hours Worked" column.

IF(ISNOTBLANK([Out Time]), ([Out Time] - [In Time]), "")

* This formula is designed to handle time calculations in AppSheet, specifically addressing scenarios where time spans across midnight. Let's break down the code and discuss where and how to use it:

IF(

ISNOTBLANK([Check-Out-DateTime]),

IF(

([Check-Out-DateTime] < [Check-In-DateTime]),

([Check-Out-DateTime] + "24:00:00") - [Check-In-DateTime],

[Check-Out-DateTime] - [Check-In-DateTime]

),

"PENDING"

)

IF(

ISNOTBLANK([Out-DateTime]),

IF(

([Out-DateTime] < [Check-In-DateTime]),

([Out-DateTime] + "24:00:00") - [Check-In-DateTime],

[Out-DateTime] - [Check-In-DateTime]

),

"PENDING"

)

1. ISNOTBLANK([Out Time]): This checks if the "Out Time" field is not blank, ensuring that the calculation is only performed when an "Out Time" value is entered.
2. IF([Out Time] < [In Time], ...: This condition checks if the "Out Time" is less than the "In Time," which indicates that the time span crosses midnight.
3. ([Out Time] + "24:00:00") - [In Time]: If the time span crosses midnight, this part of the formula adds a full day (24 hours) to the "Out Time" and then subtracts the "In Time" to correctly calculate the time difference.
4. [Out Time] - [In Time]: If the time span doesn't cross midnight, this part of the formula directly subtracts the "In Time" from the "Out Time" to calculate the time difference.
5. "": If the "Out Time" is blank, the formula returns an empty string, ensuring that no calculation is performed and the cell remains blank.

### **Usage**:

1. Time Calculations: Use this formula in scenarios where you need to calculate time differences, especially when dealing with time spans that may cross midnight (e.g., tracking working hours that start in the evening and end in the morning).
2. Data Validation: This formula can also be used for data validation to ensure that time entries are logical and correctly handled, preventing errors such as negative time differences.
3. AppSheet Applications: Incorporate this formula into your AppSheet app's virtual columns or calculated fields wherever time calculations with potential midnight crossover are required.
4. By using this formula, you can accurately handle time calculations in AppSheet applications, ensuring correct results even when dealing with time spans that cross midnight.

* This code is typically used in AppSheet to check if there is a corresponding "Out Time" entry in the "Check Out Time" table for a specific employee ("Empno") on either the same date as the current row's "Attendance Date" or the next day. This can be useful in scenarios where you want to validate attendance data or perform actions based on attendance information for employee.

Time:

any(select(Check Out Time[Out Time], and([\_THISROW].[Empno] = [Empno],

or(

[Attendance Date] = [\_THISROW].[Attendance Date],

[Attendance Date] = ([\_THISROW].[Attendance Date] + 1) ))))

Date:

any(select(Check Out Time[Attendance Date], and([\_THISROW].[Empno] = [Empno],

or(

[Attendance Date] = [\_THISROW].[Attendance Date],

[Attendance Date] = ([\_THISROW].[Attendance Date] + 1) ))))

**Breakdown**:

* any(select(...)): This part of the expression uses the SELECT function to retrieve values from a specified column in a specified table. The ANY function is then used to check if any value meets the conditions specified within the SELECT function.
* Check Out Time[Out Time]: This specifies the column "Out Time" in the table "Check Out Time" from which values will be retrieved.
* and(...): This is a logical function that checks if multiple conditions are true. In this case, it checks if both conditions within it are true.
* [\_THISROW].[Empid] = [Empid]: This compares the "Empid" value of the current row (\_THISROW) with the "Empid" value in the retrieved row from the "Check Out Time" table.
* or(...): This is another logical function that checks if at least one condition within it is true. It's used here to check if either of the conditions inside it is true.
* [Attendance Date] = [\_THISROW].[Attendance Date]: This checks if the "Attendance Date" in the retrieved row matches the "Attendance Date" in the current row.
* [Attendance Date] = ([\_THISROW].[Attendance Date] + 1): This checks if the "Attendance Date" in the retrieved row is exactly one day after the "Attendance Date" in the current row.
* The formula utilizes conditional logic to determine access permissions based on user roles and usernames. It first checks if the user's role is "Admin" by searching for this role in the "User Login Details" table where the username matches the current user's username. If the user is an admin, it returns TRUE, granting access. If the user is not an admin, it checks if the current user's username matches the username in the "Usernames" column of the same row, returning TRUE if there's a match, indicating access based on the username.

IF(IN("Admin",select(User Login Details[Role],[\_THISROW].[Username]=[Usernames])),TRUE,[\_THISROW].[Username]=[Usernames])

* FOR NOW, WORKING ID FETCHING FORMULA FOR CHECK OUT TIME “ID”COLUMN

any(

select(Check In Time[ID],and([\_THISROW].[Empno] = [Empno],[Check-Out-DateTime] = MAX(select(Check In Time[Check-In-DateTime],and([\_THISROW].[Empno] = [Empno],

or([Check-Out-DateTime] = [\_THISROW].[Check-Out-DateTime],

[Check-Out-DateTime] = [\_THISROW].[Check-Out-DateTime] - 1

)))))))

SECOND ID FETCHING FORMULA FOR CHECK OUT TIME “ID”COLUMN

any(select(Check In Time[ID],and([\_THISROW].[Empno] = [Empno],[Attendance Date] = MAX(

select(

Check In Time[Attendance Date],

and( [\_THISROW].[Empno] = [Empno], [Attendance Date] <= [\_THISROW].[Attendance Date]

))))))

New Working Formula For Fetching ID from Check In Time Table to Check Out Time Table

any(select(Check In Time[ID],and([\_THISROW].[Empno] = [Empno],[Check-In-DateTime] = MAX(

select(

Check In Time[Check-In-DateTime],

and(

[\_THISROW].[Empno] = [Empno],

[Check-In-DateTime] <= [\_THISROW].[Check-Out-DateTime]

))))))

If want to fetch any ID in a column of same table from the table itself then use this formula  
  
SELECT(Check In Time[ID],AND([Empno] = [\_THISROW].[Empno],

[Check-In-DateTime] = MAX(SELECT(Check In Time[Check-In-DateTime], [Empno] = [\_THISROW].[Empno]))

)

)

* How can I create role-based access control in an AppSheet app using data from the User Login Details table?

To achieve role-based access control in your AppSheet app based on the role column in the User Login Details table, follow these steps:

* Navigate to the AppSheet Editor:
  + Open your AppSheet app in the editor.
* Access View Settings:
  + For each view that you want to restrict based on user roles (e.g., Check In and Check Out views), go to the view settings.
* Set the Show\_If Formula:
  + In the view settings, find the "Show\_If" property.

Use the following formula for the Check In and Check Out views:

IF(IN("User", SELECT(User Login Details[Role], [Name] = USERSETTINGS("usernames"))),

TRUE, FALSE)

This formula checks if the current user's role (retrieved from the User Login Details table based on their name) is "User." If yes, it returns TRUE, allowing access to the view; otherwise, it returns FALSE, restricting access.

USERSETTINGS("User"): This function retrieves the value associated with the setting named "User" for the current user. In your case, you've set it up so that this setting holds the user's name.

Second formula: To provide access to Both Admin & Super Admin  
IF(

IN("Admin", SELECT(User Login Details[Role], [Name] = USERSETTINGS("Usernames"))),

TRUE,

IN("Super Admin", SELECT(User Login Details[Role], [Name] = USERSETTINGS("Usernames")))

)

Or SImple THIS FORMULA ALSO USED TO ACTIVATE ACTION OR BEHAVIOR TO “Admin” only.

[Role]="Admin"

* To see the data of all users by admin. THIS FORMULA ALSO USED TO ACTIVATE ACTION OR BEHAVIOR TO “Admin” only

IF([Role]="Admin",TRUE,[Username]=[Usernames])

* TO GIVE ACCESS OF ACTION TO USERS BASED ON CONDITION  
  AND(ISBLANK([Out-DateTime]),[Role] = "Admin")
* TO FILTER VIEWS IN APPSHEET USING SLICES OPTION  
  DATE([Check-In-DateTime]) >= TODAY() - 1
* TO FORMAT ENTRIES BASED ON CERTAIN CONDITION   
  FRIST FORMULA:  
  ISNOTBLANK([Out-DateTime])  
  SECOND FORMULA:

AND(ISNOTBLANK([Check-In-DateTime]),ISBLANK([Out-DateTime]))

THIRD FORMULA:

OR(DATE([Check-In-DateTime]) >= TODAY() - 1,ISBLANK([Out-DateTime])))

* How can I modify my AppSheet virtual column formulas to handle scenarios where an employee checks in on one date, checks out on another date, then checks in again on the same check-out date? Currently, the system shows a duplicate entry for the evening check-in without a proper check-out, and it fetches the previous check-out time for the current check-in. What I want is for the system to treat the evening check-in as a new entry and not fetch the previous check-out time until the employee checks out again. How can I adjust the formulas to achieve this in AppSheet?
* TO CONVERT OR EXTRACT MONTH NAME FROM A DATE COLUMN IN APPSHEET

SWITCH(MONTH([Date]),

1, "January",

2, "February",

3, "March",

4, "April",

5, "May",

6, "June",

7, "July",

8, "August",

9, "September",

10, "October",

11, "November",

12, "December",

"Invalid Month Number"

)

SECOND FORMULA: MONTH + YEAR

CONCATENATE(SWITCH(MONTH([Date]),

1, "January",

2, "February",

3, "March",

4, "April",

5, "May",

6, "June",

7, "July",

8, "August",

9, "September",

10, "October",

11, "November",

12, "December",

"Invalid Month Number"

), " ", YEAR([Date]))

**TO EXCLUDE LUNCH TIME FOR 12 pm & 1 TO 2 pm**

IF(

AND(

TIME([Out-DateTime]) > TIME("01:00 PM"),

TIME([Check-In-DateTime]) < TIME([Out-DateTime])

),

HOUR([Out-DateTime] - [Check-In-DateTime]) + MINUTE([Out-DateTime] - [Check-In-DateTime]) / 60 + (SECOND([Out-DateTime] - [Check-In-DateTime])) / 3600 - 1,

HOUR([Out-DateTime] - [Check-In-DateTime]) + MINUTE([Out-DateTime] - [Check-In-DateTime]) / 60 + (SECOND([Out-DateTime] - [Check-In-DateTime])) / 3600

)

SECOND FORMULA:

IF(

AND(

TIME([Out-DateTime]) >= TIME("13:00:00"),

TIME([Out-DateTime]) <= TIME("14:00:00")

),

HOUR([Out-DateTime] - [Check-In-DateTime]) + MINUTE([Out-DateTime] - [Check-In-DateTime]) / 60 + SECOND([Out-DateTime] - [Check-In-DateTime]) / 3600,

HOUR([Out-DateTime] - [Check-In-DateTime]) + MINUTE([Out-DateTime] - [Check-In-DateTime]) / 60 + SECOND([Out-DateTime] - [Check-In-DateTime]) / 3600 - 1)

* **WORKING FORMULA FOR SANTOSHI CONSTRUCTION SCENARIO 👍**

IF(AND(OR(TIME([Out-DateTime]) > TIME("14:00:00"),AND(

TIME([Out-DateTime]) >= TIME("13:00:00"),

TIME([Out-DateTime]) <= TIME("14:00:00")

)),TIME([Check-In-DateTime]) <= TIME("13:00:00")

),

HOUR([Out-DateTime] - [Check-In-DateTime]) + MINUTE([Out-DateTime] - [Check-In-DateTime]) / 60 + SECOND([Out-DateTime] - [Check-In-DateTime]) / 3600 - 1,

HOUR([Out-DateTime] - [Check-In-DateTime]) + MINUTE([Out-DateTime] - [Check-In-DateTime]) / 60 + SECOND([Out-DateTime] - [Check-In-DateTime]) / 3600 )

* This formula is used to fetch any details from other table in virtual column

any(select(Check Out Time[Check-Out-DT], and([\_THISROW].[Empno] = [Empno], [\_THISROW].[ID]=[ID])))

=TEXT(D20, "mm/dd/yyyy") & " " & TEXT(E20, "hh:mm:ss")

FILTER("Check In Time", ([ID] = [\_THISROW].[ID]))

* **To sum values from another table in a table based on conditions**:

SUM(

SELECT(

Check In Time[EXCLUDE LUNCH],

AND(

[Empno] = [\_THISROW].[Empno],

MONTH([Check-In-DateTime]) = [\_THISROW].[Month],

YEAR([Check-In-DateTime])=[\_THISROW].[YEAR]

)

)

)

Second formula:

SUM(

SELECT(

Check In Time[EXCLUDE LUNCH],

AND(

[Empno] = [\_THISROW].[Empno],

[MONTH] = [\_THISROW].[Month]

)

)

)

Power BI Formulas:

To calculate Work Hours:

Work Hours =

VAR StartDateTime = MAX('Check In Time'[Attendance Date]) + MAX('Check In Time'[In Time])

VAR EndDateTime = MAX('Check Out Time'[Attendance Date]) + MAX('Check Out Time'[Out Time])

VAR WorkDuration = IF(EndDateTime < StartDateTime, EndDateTime + 1 - StartDateTime, EndDateTime - StartDateTime)

RETURN

TIME(HOUR(WorkDuration), MINUTE(WorkDuration), SECOND(WorkDuration))

any(select(Check Out Time[Check-Out-DT], and([\_THISROW].[Empno] = [Empno], [\_THISROW].[ID]=[ID])))

**Looker Studio Formulas:**

* **Looker Studio** Formula to find absent employees  
  COUNT\_DISTINCT(Empid (Table 2))- COUNT\_DISTINCT(Empid (Table 1))
* If want any text if the employee is absent  
  CASE WHEN COUNT\_DISTINCT(Empid (Table 2))- COUNT\_DISTINCT(Empid (Table 1)) = 1 THEN "A" ELSE NULL END
* If want to change the result in “TEXT” format  
  CAST(COUNT\_DISTINCT(Empid (Table 2)) - COUNT\_DISTINCT(Empid (Table 1)) AS STRING)
* **Whenever there is null value & you want to do any calculation then use this approach to avoid null in calculations.**

CASE WHEN (Advance Pay) IS NULL THEN (EARNINGS) ELSE EARNINGS - Advance Pay END

**EMPNO. IN CHECK IN TIME TABLE VALID IF FORMULA WITH ERROR MESSAGE:**

THIS FORMULA IS PROPERLY WORKING AS A VALIDATION FOR THE EMPLOYEE TO CHECK IN. THIS FORMULA CHECK THE EMPNO IS VALID OR NOT, EMPLOYEE CAN NOT CHECK IN AGAIN BEFORE HIS CHECK OUT.

AND( ISBLANK(FILTER( "Check In Time",AND([Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = DATE([\_THISROW].[Check-In-DateTime]),ISBLANK([Out-DateTime]) ) ) ),

ISBLANK(FILTER("Check In Time",AND([Empno] = [\_THISROW].[Empno],ISBLANK([Out-DateTime]) ))),

NOT(ISBLANK(LOOKUP([\_THISROW].[Empno], "Employees", "Empno", "Empno"))))

**CUSTOM ERROR MESSAGE:**

IF( ISBLANK(LOOKUP([\_THISROW].[Empno], "Employees", "Empno", "Empno")),

"INVALID EMPLOYEE ID","PLEASE ENSURE THE PREVIOUS CHECK-IN IS CHECKED OUT")

**THIS ABOVE FORMULA IS PERFECT, BUT THERE IS A NEW UPDATE IN THIS BELOW FORMULA FOR EXTRA VALIDATION OPTIONS.**

**Ensuring Accurate Employee Check-In with Advanced Validation:**

The Check-In Validation feature in the AppSheet application for employee attendance ensures that employees can efficiently record their attendance while maintaining data integrity and preventing erroneous entries.

#### 

#### **Purpose:**

#### The purpose of this feature is to enforce strict rules for employee check-ins based on specific conditions, enhancing the accuracy and reliability of attendance records in the app.

#### **Feature Highlights:**

* **Date-Sensitive Check-In**: Employees are allowed to check in multiple times on the current day (TODAY()), provided they have checked out earlier in the day.
* **Prevent Duplicate Check-Ins**: Employees cannot check in more than once on any past date without completing a previous check-out.
* **Integration with Employees Table**: Ensures that only valid employee IDs (Empno) listed in the Employees table can perform check-ins.

#### **Example Scenarios:**

1. **Current Day Check-In**:
   * An employee checks in at 9:00 AM on June 28, 2024.
   * They can check in again later in the day (after a check-out) if necessary.
2. **Past Date Check-In**:
   * An employee forgets to check in on June 20, 2024.
   * Attempts to check in on June 28, 2024, without a prior check-out are blocked.
3. **Existing Employee ID Validation**:
   * Only employees with valid Empno listed in the Employees table can check in.

#### **Implementation Steps:**

1. **AppSheet Configuration**:
   * Navigate to the Data tab in AppSheet.
   * Select the Check In Time table.
   * Locate the Check-In-DateTime column.
   * Set the validation formula under "Valid If" to enforce the described conditions.
2. **Validation Formula**:

AND( ISBLANK(FILTER( "Check In Time", AND( [Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = DATE([\_THISROW].[Check-In-DateTime]),

DATE([Check-In-DateTime]) <> TODAY(),ISNOTBLANK([Out-DateTime])))),

ISBLANK(FILTER( "Check In Time",AND([Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = TODAY(),

ISBLANK([Out-DateTime]) ) )),

ISBLANK(FILTER( "Check In Time",AND([Empno] = [\_THISROW].[Empno], DATE([Check-In-DateTime]) < TODAY(),ISBLANK([Out-DateTime]) ))),

NOT(ISBLANK( LOOKUP([\_THISROW].[Empno], "Employees", "Empno", "Empno"))))

**Validation Formula:**

The formula now checks three conditions to ensure the employee can't check in under the specified scenarios.

* It ensures no check-ins on the same date (excluding today) without a check-out.
* It ensures no check-ins today without a check-out.
* It ensures no past check-ins without a check-out.
* The formula still ensures the employee exists in the Employees table (Empno exists).

**3. Validation Error Message**:

IF(ISBLANK(LOOKUP([\_THISROW].[Empno], "Employees", "Empno", "Empno") ), "INVALID EMPLOYEE ID",

IF( NOT(ISBLANK(FILTER( "Check In Time",AND( [Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = TODAY(),ISBLANK([Out-DateTime]) )))),

"PLEASE CHECK OUT FOR YOUR PREVIOUS ENTRY",

IF(NOT(ISBLANK(FILTER("Check In Time",AND([Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) < TODAY(),

ISBLANK([Out-DateTime]) )))),

"YOU HAVE AN UNFINISHED CHECK-IN FROM A PREVIOUS DATE",

IF(NOT(ISBLANK(FILTER("Check In Time",AND([Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = DATE([\_THISROW].[Check-In-DateTime]),

DATE([Check-In-DateTime]) <> TODAY(),ISNOTBLANK([Out-DateTime]))) )),

"THERE IS ALREADY A CHECK-IN RECORD ON SELECTED DATE OF THIS EMPLOYEE", ""))))

**Error Messages:**

* If the employee ID does not exist in the Employees table, it shows **"INVALID EMPLOYEE ID".**
* If the employee has already checked in today without checking out, it shows **"PLEASE CHECK OUT FOR YOUR PREVIOUS ENTRY".**
* If the employee has an unfinished check-in from a previous date, it shows **"YOU HAVE AN UNFINISHED CHECK-IN FROM A PREVIOUS DATE".**
* **THERE IS ALREADY A CHECK-IN RECORD ON SELECTED DATE OF THIS EMPLOYEE**: Shown if the employee tries to check in twice on the same day EXCEPT TODAY() DATE

**TO SET-UP RESTRICTION FOR PAYMENT FROM CLIENTS**

To accommodate the initial login view and integrate the payment restriction, we need to modify the initial view logic to first check the payment status. If the payment is not made, users will be redirected to the payment view. Once the payment is verified, users can proceed to the usual login view.

**1. Payments Table**

Ensure you have a `Payments` table with the following columns:

- `ID` (Auto-generated key)

- `Payment Date` (Date)

- `Paid By` (Ref to Users table, referencing Prateek)

- `Amount` (Number)

- `Payment Status` (Enum with values: "Paid", "Unpaid")

- `Payment Link` (URL)

**2. Add a Virtual Column in Payments Table**

Create a virtual column named `Is Payment Current` with the following formula:

IF(AND( [Paid By] = "Prateek", [Payment Date] >= EOMONTH(TODAY(), -1) + 1,

[Payment Date] <= EOMONTH(TODAY(), 0), [Payment Status] = "Paid" ), "Yes", "No")

**3. Restriction Slice**

Create a slice that filters users based on the payment status.

1. Go to Data -> Slices -> + New Slice

2. \*\*Name\*\*: `AccessRestriction`

3. \*\*Source Table\*\*: Your main user table or any table that needs to be restricted.

4. \*\*Row Filter Condition\*\*:

OR( USEREMAIL() = "prateek@example.com", IN("Yes",SELECT(Payments[Is Payment Current], [Paid By] = "Prateek") ))

**4. Restriction View**

Create a view that informs users about the payment restriction and prompts them to make the payment.

1. Go to UX -> Views -> + New View

2. View Name: `PaymentRestriction`

3. For This Data: Any table, but you can create a dummy table if needed.

4. View Type: `Detail`

5. Show If:

NOT(IN( "Yes", SELECT(Payments[Is Payment Current], [Paid By] = "Prateek") ))

**6. Content: Add text or instructions for making the payment.**

**5. Modify Initial View Logic**

Configure the app to show the restriction view if the payment is not made and redirect to the usual login view if the payment is verified.

1. Go to UX -> Options

2. Starting View:

IF(NOT(IN("Yes",SELECT(Payments[Is Payment Current], [Paid By] = "Prateek")) ),

LINKTOVIEW("PaymentRestriction"),LINKTOVIEW("UserSettings"))

**6. Modify Existing Access Control**

Ensure all views have the correct access control logic in addition to the payment status check. Here’s how you can combine both conditions:

**For Each View in Your App:**

1. Go to UX -> Views

2. Select the view you want to configure.

3. In the `Show If` condition, combine your existing access control logic with the payment status check.

**For example:**

AND(OR(

[UserRole] = "Admin",

[UserRole] = "Manager",

[UserRole] = "Employee"

), IN("Yes", SELECT(Payments[Is Payment Current], [Paid By] = "Prateek")))

**Example Access Control Logic:**

**Admin Views: Only accessible by Admins and if payment is made.**

AND(

[UserRole] = "Admin",

IN("Yes",SELECT(Payments[Is Payment Current], [Paid By] = "Prateek") ) )

**Manager Views:** Only accessible by Managers and if payment is made.

AND(

[UserRole] = "Manager",

IN("Yes",SELECT(Payments[Is Payment Current], [Paid By] = "Prateek") ))

**Employee Views: Only accessible by Employees and if payment is made.**

AND(

[UserRole] = "Employee",

IN( "Yes",SELECT(Payments[Is Payment Current], [Paid By] = "Prateek") ) )

**7. Create an Action to Redirect to Payment Form**

Create an action to redirect Prateek to the payment form if the payment is due.

1. Go to Behavior -> Actions -> + New Action

2. For a Record of This Table: `Payments`

3. Do This: `App: go to another view within this app`

4. Target: LINKTOFORM("Payment Form", "Paid By", "Prateek")

5. Display Name: `Make Monthly Payment`

6. Icon: Choose an appropriate icon.

7. Only if this condition is true:

AND(

USEREMAIL() = "prateek@example.com",

NOT(IN("Yes",SELECT(Payments[Is Payment Current], [Paid By] = "Prateek"))))

**8. Add the Action to the User Interface**

Add the action to a view where Prateek can see it easily.

1. Go to UX -> Views

2. Select a view where Prateek can see the action, such as a dashboard or home screen.

3. Scroll down to the `Actions` section within the view configuration.

4. In the `Behavior` area, look for `Event Actions`.

5. Under `Row Selected`, select `Make Monthly Payment`.

6. Click `Save` to apply the changes.

**Summary**

1. \*\*Payments Table\*\*: Track payment status.

2. \*\*Virtual Column\*\*: Check if the current month's payment is made.

3. \*\*Restriction Slice\*\*: Control access based on payment status.

4. \*\*Restriction View\*\*: Inform users about payment restriction.

5. \*\*Initial View\*\*: Set starting view based on payment status.

6. \*\*Modify Access Control\*\*: Combine payment status check with existing access control logic.

7. \*\*Create Action\*\*: Redirect Prateek to the payment form.

8. \*\*Add Action to UI\*\*: Make it visible for Prateek.

By following these steps, users will be restricted from accessing the login view and using the app until Prateek makes the payment for the current month. Once the payment is made, users can log in and use the app as usual, preserving the existing access control settings.

**New formula to avoid duplicate entries in attendance app or if the employee id is not available in the employees table**  
AND(ISBLANK(FILTER("Check In Time",AND([Empno] = [\_THISROW].[Empno],

DATE([Check-In-DateTime]) = DATE([\_THISROW].[Check-In-DateTime]),

ISBLANK([Out-DateTime]) ) )),

ISBLANK( FILTER( "Check In Time", AND( [Empno] = [\_THISROW].[Empno],

ISBLANK([Out-DateTime])

))),

NOT(ISBLANK( LOOKUP([\_THISROW].[Empno], "Employees", "Empno", "Empno"))))

Formula for error message  
IF( ISBLANK(LOOKUP([\_THISROW].[Empno], "Employees", "Empno", "Empno")),

"INVALID EMPLOYEE ID","PLEASE CHECK OUT, FOR YOUR PREVIOUS ENTRY")