WUNDERMAN THOMPSON

AMPscript for Intermediate/Advanced Users

Wunderman Thompson Salesforce Center of Excellence

AGENDA

01 **Taking the Next Steps**

02 Working with Rows

03 Expanding Your Toolkit

04 Seeing it in Action

05 Q&A

- You know the basics, now what?

AGENDA

01 Taking the Next Steps

02 Working with Rows

03 Expanding Your Toolkit

04 Seeing it in Action

05 Q&A

- How to lookup and return a full row of data?
- How to get to the data you actually need to use?
- What to do with multiple rows?

AGENDA

01 Taking the Next Steps

02 Working with Rows

03 Expanding your Toolkit

- What else can you do with AMPscript?
- Lesser known functions.
- Programming defensively.

04 Seeing it in Action

05 Q&A

AGENDA

01 Taking the Next Steps

02 Working with Rows

03 Expanding Your Toolkit

04 Seeing it in Action

05 Q&A

- Taking a look at a practical example.

AGENDA

01 Taking the Next Steps

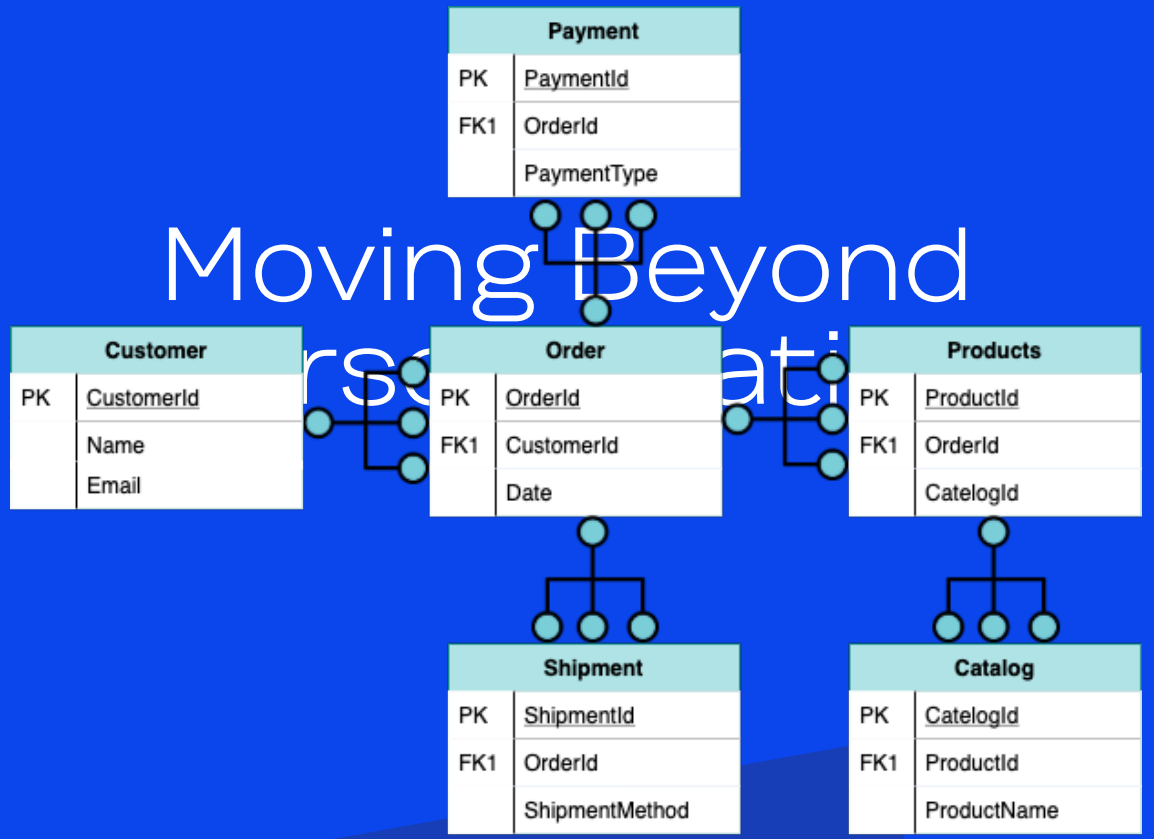
02 Working with Rows

03 Expanding Your Toolkit

04 Seeing it in Action

05 Q&A

Work with relational data and arrays.

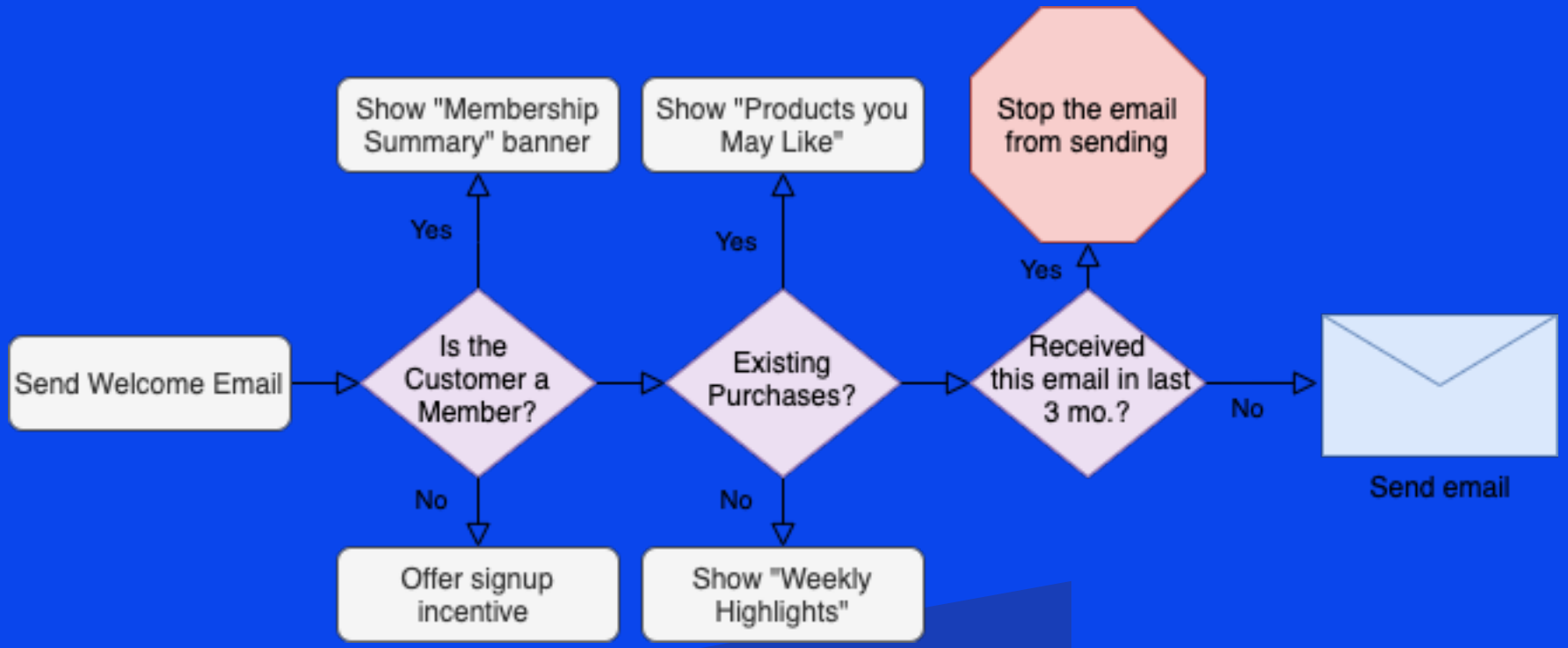


Moving Beyond

WUNDERMAN THOMPSON

Moving Beyond Personalization

Implement conditional logic to control what happens and when.



Moving Beyond Personalization

Leverage SOAP and REST APIs to interact with other Marketing Cloud Components and external systems.

Working with Rows

What happens when your data is not just a field, but a whole row?

Working with Rows

Retrieving Rows

There are two common functions used for retrieving rows of data from a data extension:

LookupRows – Returns a rowset (array) of records from a data extension that meets the specified values.

LookupOrderedRows – Returns a rowset (array) of records in a specified order and quantity.

*Note - there are case sensitive (CS) variations of these functions if needed.

LookupRows Function Syntax

LookupRows

Syntax

LookupRows(1, 2, 3)

1. Name of data extension from which to return specified rows.
2. Column name used to identify rows to return.
3. Value used to match rows to return.

Example

```
%%[
SET @OrderRows = ‘LookupRows(‘OrderItems’, ‘OrderId’, @OrderId)
]%%
```

Returns

A rowset or records from the “OrderItems” data extension where “OrderId” matches the value of the @OrderId variable.

LookupRows Function Syntax

Working with Rowsets

FOR Process Loops

Syntax:

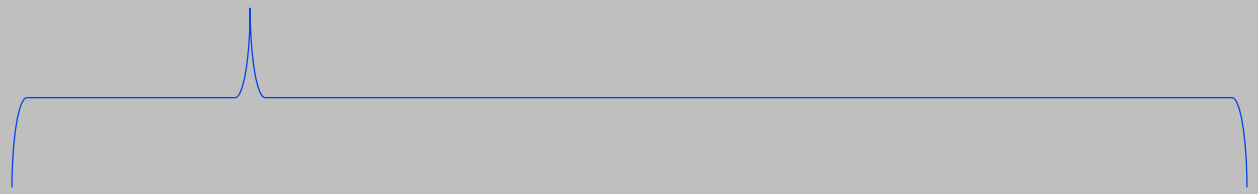
```
%%[
    FOR @iterator = <start> TO <end> DO
]%%
        <execute script or content>
%%[
    NEXT @iterator
]%%
```

LookupRows Function Syntax

Working with Rowsets

FOR Process Loops

FOR @iterator = <start> TO <end> DO



The FOR keywork initiates the process loop.

LookupRows Function Syntax

Working with Rowsets

FOR Process Loops

FOR @iterator = <start> TO <end> DO

The iterator variable can be called anything and contains the integer of the iteration of the loop. This starts at 1.

LookupRows Function Syntax

Working with Rowsets

FOR Process Loops

FOR @iterator = <start> TO <end> DO

The <start> is an expression to define what number of the @iterator should be used as the starting point for the loop, which can be an integer or variable. To start at the beginning, this would be 1.

LookupRows Function Syntax

Working with Rowsets

FOR Process Loops

FOR @iterator = <start> TO <end> DO

Much like the start expression, the <end> expression specifies when the loop iterations should stop. This can be an integer or a variable.

For example, to stop at the completion of iterating through all records of a rowset:

<end> = RowCount(@OrderRows)

LookupRows Function Syntax

Working with Rowsets

FOR Process Loops

NEXT @iterator

Using the NEXT keyword and your variable for the iterator triggers the loop to check if the current iteration is after the end expression or if not, the loop should repeat.

Field and Row Functions

Field and Row Functions

Row

Extracts the specified row from a rowset

Syntax

Row(1, 2)

1. The specified rowset to obtain the row from.
2. The row number of the row to return.

Example

%%[SET @Row = Row(@OrderRows, 1)]%%

Field and Row Functions

Field

Returns the specified field in the specified row

Syntax

Field(1, 2, 3)

1. Row from which to return the field.
2. Name of the field or attribute to return.
3. (optional) Indicates whether to return a NULL value or an error if the specified data extension field does not exist

Example

%%[SET @Product = Field(@Row, 'ProductName')]%%

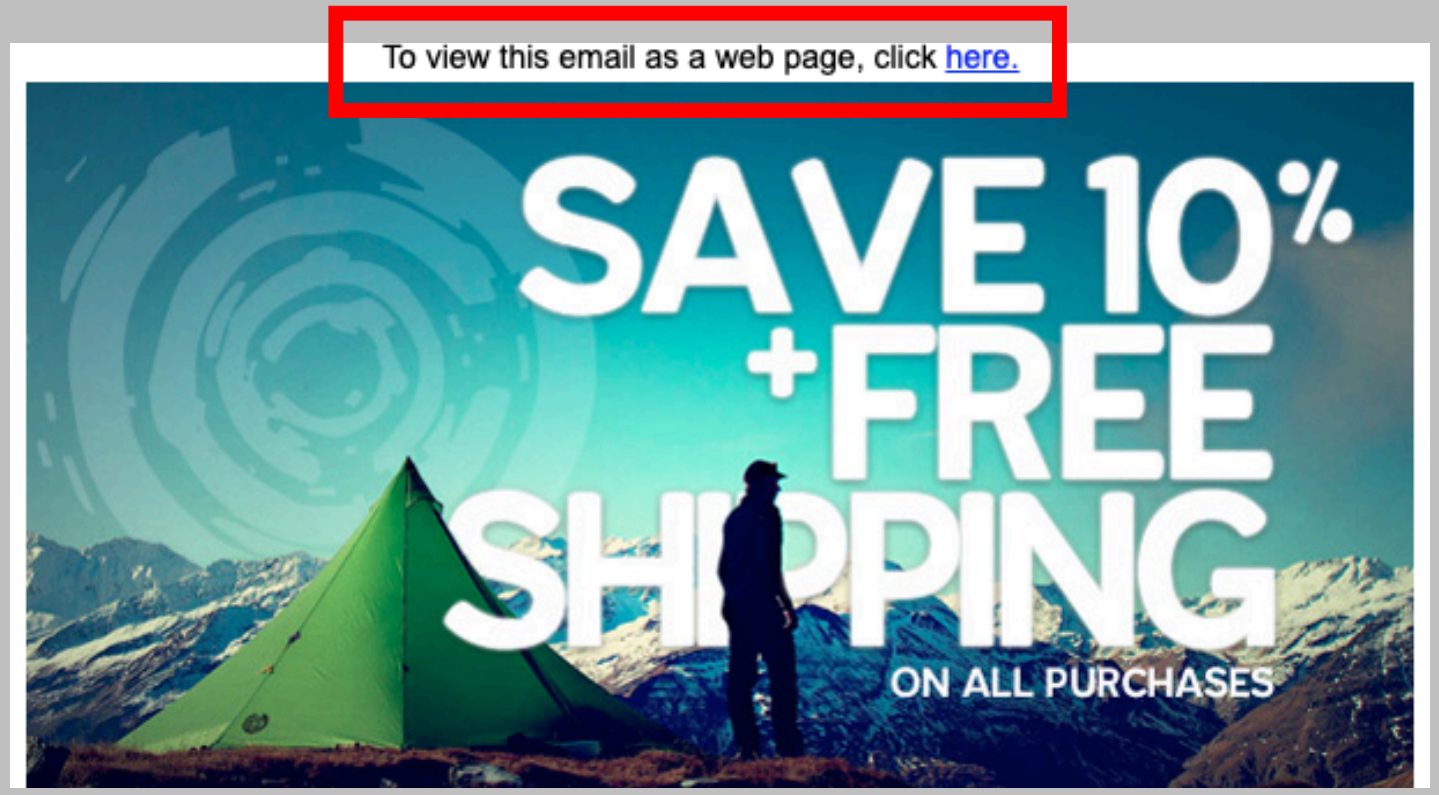
Putting it All Together

1. Define your rowset
2. Start your loop
3. Set your fields from the row
4. Add your content
5. Close the loop

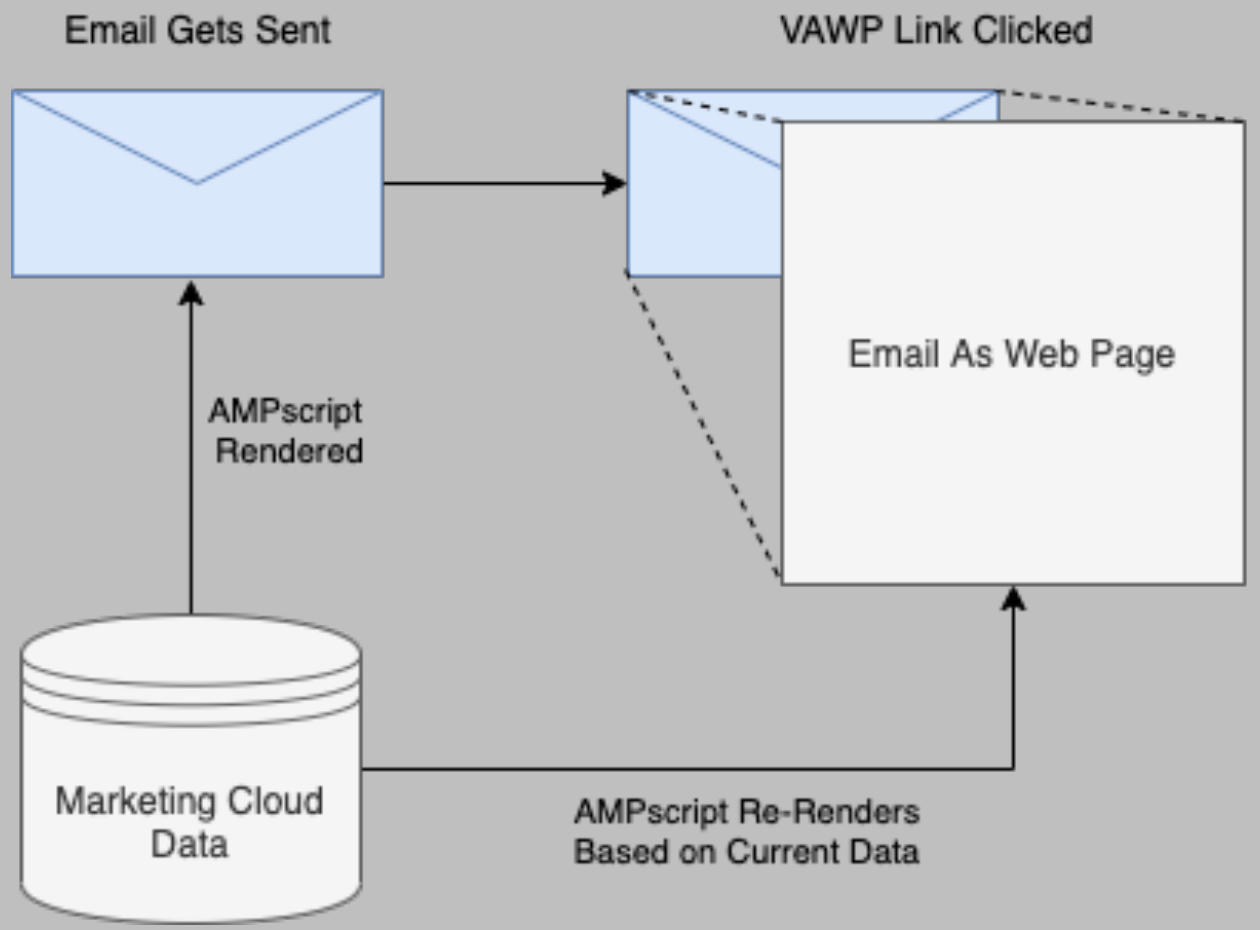
```
%%[
        SET @OrderId = OrderId
        SET @OrderRows =
LookupRows('OrderItems',    'OrderId',
@OrderId)
        FOR @i = 1 TO
RowCount(@OrderRows) DO
            SET @Product =
  Field(Row(@OrderRows, @i),  'ProductName')
            SET @Price =
  Field(Row(@OrderRows, @i),  'Price')
            SET @Quantity =
  Field(Row(@OrderRows,       @i), 'Quantity')
  ]%%

    <HTML and AMPscript content to display>

  %%[   NEXT @i   ]%%
```

Tackling Different Use Cases

Dealing with View as a Web Page (VAWP)



Dealing with View as a Web Page (VAWP)



Dealing with View as a Web Page (VAWP)

%%_messagecontext%%

Values include:

- **SEND** - Display the rendered final message for sending to subscriber
- **PREVIEW** - Display the send preview options available within editor
- **VAWP** - Display content
- VIEWSENT - Display the non-subscriber link to preview content
- FTAF - Display the rendered Forward To a Friend message
- LANDINGPAGE - Display a landing page or microsite
- VALIDATION - Display information corresponding to the validate option in Marketing Cloud
- LINKRESOLUTION - Display resolved dynamic script at click time
- SMS - Display SMS message content
- SOCIAL - Display Social Forward content
- SITE - Display CloudPage content

Dealing with View as a Web Page (VAWP)

%%_messagecontext%%

1. Set up a send log / data archival process.

Dealing with View as a Web Page (VAWP)

%%_messagecontext%%

2. Write an IF statement using _messagecontext values.

```
%%[
IF _messagecontext == 'VAWP' THEN
        SET @Fname = Lookup('SendLog', 'FirstName',
        'SubscriberKey', _subscriberkey, 'EmailName',
        emailname_)
ELSE
        SET @Fname = FirstName
ENDIF
]%%
```

Using Data Views in AMPscript

_Sent

Contains records for all email sends to subscribers over the last 6 months.

_Job

Provides details about the send activity, such as email name, from address, and subject line.

_Open

Shows records for every email open that occurs.

_Click

Records all click information from your emails, including URL, alias, etc.

HTTP Functions

Need to get content or data from external platforms?

HTTP functions allow you to make GET (retrieve) and POST (send) calls to a specified URL.

HTTP Functions

Some possible use cases:

1. You need to pull content from an RSS feed into your email. (HTTPGet)
2. An external system generates unique coupon codes as requested and you want to display it in the email. (HTTPGet)
3. You want to send an update to an external system to generate follow-up activities based on the email send. (HTTPPost)

Programming Defensively

Programming Defensively

Step 1: Know Your Data

Is the field I need required?

What is the data type that I'm working with?

How is the data formatted?

Programming Defensively

Step 2: Proactive Logic

Know your functions

Use IF Statements and set fallback values

Convert data types as needed

Use RaiseError() when all else fails

Use Server-Side JavaScript try/catch logic if desired*

*Requires additional programing for SSJS syntax.

Programming Defensively

Step 3: Log Errors and Test

Create error log data extension(s)

Setup scheduled resend jobs

Create representative test lists

WUNDERMAN THOMPSON

Thank You

BLACK LIVES MATTER

HAPPY PRIDE MONTH

Salesforce Center of Excellence