RESERVE BANK OF AUSTRALIA

RBAFOI-262728

An application was received seeking access to documents as detailed below.

Summary of Request	Documents between 1 November 2019 and 25 September 2026, concerning the following: 1. The published monthly average yield of 90-day Bank Accepted Bills for each month in the period, as published. It is sufficient to identify the publication and the web address of the table in which each value appears; the values need not be re-supplied in any other form. 2. The documents held by the Bank that record the method by which that monthly average yield is compiled: the definition of the series, the inputs, the averaging convention, the rounding and publication conventions, and the publication timetable. Current versions, and any version in force during the period. 3. The documents recording the arrangements under which the Bank obtains the source data for the series, including any data supply agreement, licence or standing arrangement with the provider of the underlying market data. 4. The documents recording any supply of the series, or of any value in it, to the Australian Taxation Office, or any correspondence with that agency about the series or its use, including any standing arrangement for its supply. 5. The documents recording any correction, restatement, revision or republication of a published value in the period, and the record of the decision to make it.
Notes/context to any specific documents released:	The following links were also released to the applicant: https://www.rba.gov.au/statistics/tables/xls/f01hist.xlsx https://www.rba.gov.au/statistics/tables/changes-to tables.html

RESERVE BANK OF AUSTRALIA

Domestic Markets – Daily Processes FM Data Management User Manual

January 2026

Version Control

Version	2.7
Date	9 January 2026
Document Approver	Domestic Markets – Business Data Lead
Document Administrator	Domestic Markets – Data Management Team
Date Next Review Due	June 2026

Document Change History

Version	Status	Date	Comments
1.0	Complete	1/11/2017	Initial document containing instructions to complete tasks within Domestic Markets
1.1	Complete	20/06/2018	Review of processes and delete redundant processes
1.2	Complete	16/07/2018	Separate daily tasks from combined user manual
1.20	Complete	31/01/2019	Review completed
2.0	Complete	14/01/2022	Review completed, code links to TFS versions
2.1	Complete	01/06/2022	Review completed, F2 moved to the EDO
2.2	Complete	25/08/2022	'Daily Portfolio Summary' task moved to 'Ceased tasks' section.
2.3	Complete	25/10/2022	Process diagram for DM SMS Equities morning briefing added into the document
2.4	Complete	28/09/2023	Updates made to reflect changes in morning briefing updates and stat table F01
2.5	Complete	09/04/2024	Updates to the SMS morning briefing process.
2.6	Complete	11/09/2025	Major updates to SMS morning briefing process.
2.7	Complete	09/01/2026	Review and updating of tasks. Moving the morning briefing task into the "Automated Tasks" section.
2.8	Complete	28/07/2026	Updated hyperlink to Publication Dropbox, added a section on the basic flow of DM Morning Briefing updates.

Contents

1. Introduction **5**

2. Roles and Responsibilities **5**

3. Instruction Objectives **5**

4. Stat Table F01 **6**

5. Stat Table A03 **6**

6. Automated Tasks **8**

6.1 Morning Briefings – Equities 8

6.2 InBrief updates 8

7. Instruction Management **10**

7.1 Implementation 10

7.2 Monitoring and review 10

8. Resources **10**

8.1 Related internal documents 10

1. Introduction

The purpose of this document is to keep a track of all processes involved in successfully completing various daily tasks undertaken by Domestic Market's Data Management Team (DMDM). This user manual outlines step by step procedures involved in completing tasks related to data updates, graph updates and other ad hoc tasks completed on a daily basis.

2. Roles and Responsibilities

FM Data Management

Financial Markets Data Management Team

mailto: FMDataManagement@rba.gov.au

3. Instruction Objectives

Goal: ensure that an accurate and efficient work process is followed at all times

Sub-goals: Ease of use

Prerequisites: Certain licences will be needed in order to complete some processes

Unwanted outcomes: Errors in outputs and malfunctions in processes

4. Stat Table F01

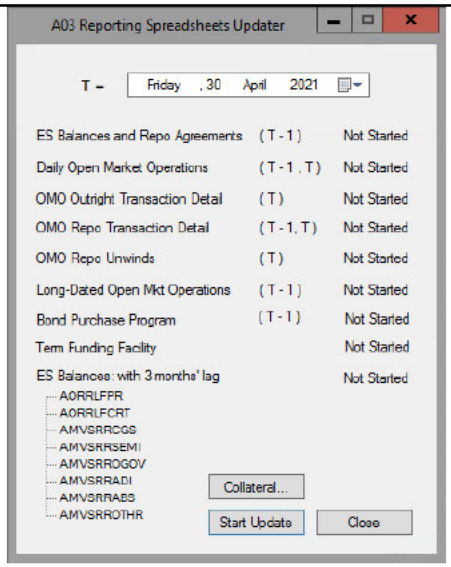
Prerequisite: To perform the F01 update, you MUST have R installed and packages required. You must have access to the Findur_R and IBOC databases.

Step	Instructions
Step 1	Note: F01 cannot be processed correctly until IBOC transaction finalises. It is typically finalised by 8:00 a.m. It is often longer when expert judgement is required. Wait for an "IBOC transactions" email with one of either: • "The was sufficient volume" or "There was insufficient volume,..., expert judgement required". • Open the f01.R and run the code
Step 2	• Check row ten is current business day. • Check the cursor in cell A1 and all data for previous business day and TRI for current business day is there.
Step 3	• Check today's 'IBOC Transactions' email to see if the Market Operations AM manager has indicated whether the cash rate will be set based on expert judgement. If not, proceed to **Step 5.**
Step 4	• If the cash rate will be set based on expert judgement, check the second email from the AM manager with approval and instructions. Typically the 'Use of Expert Judgement' tab will need to be updated for the previous business date in column A, and Judgement type 'A' in column B (occasionally type 'C') • Check the approval email to confirm the rate for column D of the 'Data' tab (same as the previous day if type 'A'). Confirm with the AM manager if this is different to their confirmation email.
Step 5	• Click 'File' and 'Save as' and save as 'xlsx', and close.
Step 6	• Drop f01d.xlsx (located here) into the Publication Dropbox

5. Stat Table A03

Note: This can only be completed after the first round of dealing, approximately 10:15am and after receiving an email confirming "DM Risk Calculation Completed" and an email confirming the OMO 1st Round FM Business Systems Health Check was successful.

Step	Instructions
Step 1	Open s 47E(d) \DM_A03Reporting.exe via file explore and wait for the pop-up window to appear.
Step 2	• The window below should pop up

	 T = should be the current day
Step 3	• Click *Start Update*. This will open A03.xls, and begin populating the table with new data. **If an error message such as** 'ES Balances must contain reporting date and reporting date – 1' open A03.xls, first tab and drag down the dates column to the next 2 weeks (weekdays only) ahead • On average it takes ~2 minutes to complete. It may take ~10 minutes on the last business day of the month when 'ES Balances: with 3 months' lag' is processed. • If there is a [Read-Only] label at the top of the excel window, it means that someone has the file opened and wont fully work. To reduce the risk of data loss, let the calculation finish. • If there was a public holiday, you may need to delete these dates from the first tab otherwise the incorrect rows may get populated. •
Step 4	• Once the table has finished updating, click *Close* in the pop-up window.
Step 5	• Click 'File' and save 'a03.xlsx', and close. • Open the Publication Dropbox on one screen, and open the MPID A03 folder in the other. Drop A03 into the Publication Dropbox by dragging and dropping.
Step 6	• You may get an email from someone in MO asking to fix some numbers: • RM PRC maintain a Minimum ES Balances for After-hours Payments table, sometime new limits are requested during the day, the table was updated to reflect them. Unfortunately, rather than creating new limits with their own effective dates, the a03 updates were made to existing limits (which changed the past). This would cause a discrepancy in Col C of 'ES Balances and Repo Agreements' sheet. • To do this, open the file like normal, and fix the cell(s) requested. • Save and close. • Reply to the person who sent the 'Approval to Publish' email asking them to remove the current version in the dropbox so you can redrop. Once removed, simply redrop.
	•

6. Morning Briefings

The morning briefing process is fully automated on the King PC. The sequence below outlines the expected emails and their approximate arrival times. If all processes run successfully and the emails show no errors, there's nothing you need to do. If any error occurs, follow the troubleshooting guide to investigate and resolve the issue.

If errors persist and updates are delayed, notify SMS at dmsms@rba.gov.au by 8:15AM so they can make alternative arrangements if required.

Note that the daily F01 statistical table is also part of the morning briefing process.

Time	Email Subject	Task Description
7:00AM	BBSW Data Update	Updates a BBSW daily data csv file
~ 7:38 AM or 7:50 AM	SMS FI Morning Briefing Service: F3	Updates F3 pricing series each morning for Fixed Income morning briefing
7:40AM	SMS EDW Data Check	Send out an email that contains a list of SMS database updates requests, in a PASS/FAIL format.
7:47AM	SMS Morning Briefing - Equities Graphs	Creates SMS equities graphs for morning briefing.

7. Automated Tasks

7.1 Morning Briefings – Equities

The morning briefing process is fully automated on the King PC. The steps below are for running the process manually if needed.

Step	Instructions
Step 1	• You will need to have the Equities-Master repo cloned to your machine. • Open and run the script FMDM_equities_morning_briefing_trycatch.R - Repos • It will run multiple scripts in parallel, and send out the email. • Data within these scripts should be querying the EDW.
Step 2	• If there are issues with any of the scripts, it will show in the console. • You will need to troubleshoot if there are issues.

7.2 InBrief updates

This is automated on the shared PC for 8:50am 11:50am weekdays

Step	Instructions

Step 0	• Make a copy of the inbrief folder, somewhere local (e.g. Documents), or clone the repository (s 47E(d) FMG.SMS/_git/inbrief). Otherwise, errors may occur. • Install any extra packages (e.g. "magick", "zip")
Step 2	• Open the 'inbrief.Rproj' and run 'run.R' from where you copy the folder to
Step 3	• A new folder should open: 'inbrief/site' • Open s 47E(d) @SSL\communities\Dashboards
Step 4	• Copy all from 'inbrief/site' and paste into s 47E(d) @SSL\communities\Dashboards • Select 'Replace the files in the destination'

8. Instruction Management

8.1 Implementation

The FM Data Management Team is responsible for the implementation of these instructions.

8.2 Monitoring and review

These instructions are reviewed by the FM Data Management Team at least quarterly or more frequently if there is a major change. All changes to the Instruction must be approved by the Senior Data Analyst.

9. Resources

9.1 Related internal documents

TRIM Link	Comments
D18/30246	Domestic Markets Data Management Back-up and Troubleshooting Manual

```
# Script to update daily statistical table: F1 Interest Rates and
Yields - Money Market ----

library(dplyr)

## Check IBOC to confirm latest finalised transactions ----
Trigger <- bankRun::sql_run(query = "SELECT TOP 1 *

                                     FROM <SESSION TABLE>

                                     WHERE IsFinalised = 1

                                     ORDER BY ModifiedAt DESC",
                            database = "<INTERNAL_DATABASE>")


## If message is the current business day then IBOC transactions
are final and f01d.xlsx can be updated
Max_Date <- Trigger$LastAllowedPublishDate

# Set Current and Previous dates for SQL queries below
Previous_Date <- Trigger$Date

if (is.na(Max_Date)) {
  stop("Query response is empty - check your IBOC access")
} else if (weekdays(as.Date(Max_Date)) == "Saturday") {
  Current_Date <- as.Date(Max_Date) + 2
} else if (weekdays(as.Date(Max_Date)) == "Sunday") {
  Current_Date <- as.Date(Max_Date) + 1
} else {
  Current_Date <- Max_Date
}

## Get data ----
### Target Cash Rate ----
DataTR <- bankRun::sql_run("

                SELECT Round(Target, 2) AS Target
                FROM <TARGET CASH TABLE>
                WHERE Date = (

                    SELECT MAX(Date)
                    FROM <TARGET CASH TABLE>
                    WHERE CONVERT(date, Date) < CONVERT(date,
GETDATE())

                    )
                ",
                           database = "<INTERNAL_DATABASE>") |>
as_tibble()
```

```
### Interbank Rate ----
SQLInterbank <- paste("
                      SELECT Date, Round(Rate, 2) AS Rate
                      FROM <INTERBANK TABLE>
                      WHERE Date = ' ", Previous_Date,"'")

DataInterbank <- bankRun::sql_run(SQLInterbank, database =
"<INTERNAL_DATABASE>")|>
  as_tibble()

### Min/max rates, Volume, Transactions and Counterparties ----
DataPublishedBaseData <-
  bankRun::sql_run("

                          SELECT s.date, p.maximumrate,
p.minimumrate,
                          round(p.totalvolume / 1000000, 0) as
Totalvolume,
                          p.numberoftransactions,
ROUND(T.TotalReturnIndex,6) AS TRI,
                          count(payer) as Counterparties

                          FROM
                          (
                            SELECT
                            t.dealDate, I.ID as Payer--,
T.Amount, T.Rate
                            FROM <TRANSACTIONS TABLE> t
                            INNER JOIN <INSTITUTIONS TABLE> I on
I.ID = T.PayerID
                            and t.IsIboc1Transaction = '1'

                            UNION
                            SELECT
                            t.dealDate, N.ID as reciever--,
T.Amount, T.Rate
                            FROM <TRANSACTIONS TABLE> t
                            INNER JOIN <INSTITUTIONS TABLE> N on
N.id = T.ReceiverID

                          WHERE t.IsIboc1Transaction = '1'

                          ) as Counttable

                          FULL JOIN <SESSION TABLE> s on s.date =
counttable.dealdate

FULL JOIN <INDEX TABLE> T on T.IBOCSession_Id = s.Id
                          FULL JOIN <PUBLISHING TABLE> p on
p.ibocsession_id = s.id
```

```
                            FULL JOIN <REPORTING VIEW> R on
R.sessiondate = s.date

                            GROUP BY date,  p.MinimumRate,
p.maximumrate, p.numberoftransactions,
                            p.totalvolume, T.TotalReturnIndex

                            ORDER BY s.date

                             ", database = "<INTERNAL_DATABASE>")|>
as_tibble()

DataPublishedData <- DataPublishedBaseData|>
  filter(date == Previous_Date)

### Adjust related column E(5) F(6) G(7) H(8)
if (
  #If transactions < 3 or counterparties < 4
  DataPublishedData["numberoftransactions"] < 3 ||
DataPublishedData["Counterparties"] < 4) {
  DataPublishedData["maximumrate"] <- "N/A"
  DataPublishedData["minimumrate"] <- "N/A"
  DataPublishedData["numberoftransactions"] <- "N/A"
}

### Forward TRI from SQL ----
DataTRI <- DataPublishedBaseData|>
  filter(date == Current_Date)

### Bank accepted bills from SQL for 1m, 3m and 6m ----
SQLBankBills <- paste("
                        SELECT Date, Maturity, Rate=ROUND(Rate, 4)
                        FROM <BANK BILLS TABLE>
                        WHERE Date = ' ", Previous_Date,"'
                          AND (TYPE = 'PM')
                          AND (Maturity IN ('1m', '3m', '6m'))
                      ") # There is an AM an PM dataset - if one
doesn't work, change Type to the other

DataBankBills <- bankRun::sql_run(SQLBankBills, database =
"<INTERNAL_DATABASE>")|>
  as_tibble()|>
  tidyr::spread(Maturity, Rate)


# Deal with the case of DataBankBills returning no record on the
day
if (count(DataBankBills) == 0) {
  DataBankBills <- data.frame(NA, NA, NA, NA)
}
```

```
### Indexed swaps from SQL and pivot ----
SQLIndexSwaps <- paste("
                        SELECT Date, Maturity, Rate=ROUND(Mid, 4)
                        FROM <OIS TABLE>
                        WHERE Date = ' ", Previous_Date,"'
                          AND DATEPART(HOUR, TIME) = 16
                          AND DATEPART(MINUTE, TIME) = 30
                          AND Maturity in ('1m', '3m', '6m')
                       ")


DataIndexSwaps <- bankRun::sql_run(SQLIndexSwaps, database =
"<INTERNAL_DATABASE>")|>
  as_tibble()|>
  # Blank the three column as not these values publish those
values currently
  mutate(Rate = NA)|>
  tidyr::spread(Maturity, Rate)

DataTreasuryNote <- data.frame(survey_term = c(1, 2, 3, 4), mid =
c(NA, NA, NA, NA))
# Transpose data frame
DataTreasuryNote <- as.data.frame(t(DataTreasuryNote))

RODBC::odbcCloseAll()

## Update f01d.xlsx ----
# Get F01 published data part to be able to edit
F01_Body <- readxl::read_xlsx('<INTERNAL_PATH>/f01d.xlsx', sheet
= 'Data',
                              skip = 10, col_types = c("date",
      "numeric", "numeric", "numeric", "numeric", "numeric",
"numeric", "numeric",
      "numeric", "numeric", "numeric", "numeric", "numeric",
"numeric", "numeric",
      "numeric", "numeric", "numeric"
    )
  )

series_ID <- colnames(F01_Body)

# Edit date and time zone to be able to navigate with date
extracted from SQL
F01_Body$`Series ID` <- as.Date(F01_Body$`Series ID`, format =
"%d-%b-%Y")

# Last target cash rate
TC <- tail(F01_Body$FIRMMCRTD, n = 2)[1]

Diff <- round(DataTR$Target - TC, 2)
```

```
if (Diff == 0) {
  Diff <- NA
}


# Put together the latest data as data frame
df <-
  data.frame(
    Previous_Date,
    DataTR$Target,
    Diff,
    DataInterbank$Rate,
    DataPublishedData[, 2:6],
    DataBankBills[, 2:4],
    DataIndexSwaps[, 2:4],
    DataTreasuryNote[2, 2:4]
  )

colnames(df) <- series_ID

dfTRI <- data.frame(Max_Date,
                    NA, NA, NA, NA, NA, NA, NA, DataTRI$TRI,
                    NA, NA,NA, NA, NA, NA, NA, NA, NA)

colnames(dfTRI) <- series_ID

# Match the date of latest data with current date series, if it
exists populate the data, if it doesn't exist add the whole row.
if (!is.na(match(df$`Series ID`, F01_Body$`Series ID`))) {
  row.to.change <- match(df$`Series ID`, F01_Body$`Series ID`) +
11
} else {
  row.to.change <- nrow(F01_Body) + 2
}

# Create data frame of the publish date
pubdate <- lubridate::today()

rpub <-
  data.frame( "Publication date",
    pubdate, pubdate, pubdate, pubdate, pubdate, pubdate,
pubdate, pubdate, pubdate,
    pubdate, pubdate, pubdate, pubdate, pubdate, pubdate,
pubdate, pubdate
  )

Sys.setenv(JAVA_HOME = '<LOCAL_PATH>System\\Files\\Java\\Windows-
x86-64')
# Load Template
wb <- xlsx::loadWorkbook("<INTERNAL_PATH>/f01d.xlsx")
```

```
ws <- xlsx::getSheets(wb)
sheet <- ws[[1]]

# Create styles
cs0 <- xlsx::CellStyle(wb) +
  xlsx::Font(wb, name = "Arial", heightInPoints = 9)

cs1 <- xlsx::CellStyle(wb) +
  xlsx::DataFormat("dd-mmm-yyyy") +
  xlsx::Alignment(wrapText = TRUE, horizontal = "ALIGN_RIGHT") +
  xlsx::Font(wb, name = "Arial", heightInPoints = 9)

cs2_6 <- xlsx::CellStyle(wb) +
  xlsx::DataFormat("0.00") +
  xlsx::Alignment(wrapText = TRUE, horizontal = "ALIGN_RIGHT") +
  xlsx::Font(wb, name = "Arial", heightInPoints = 9)

cs7_8 <- xlsx::CellStyle(wb) +
  xlsx::DataFormat("0") +
  xlsx::Alignment(wrapText = TRUE, horizontal = "ALIGN_RIGHT") +
  xlsx::Font(wb, name = "Arial", heightInPoints = 9)

cs9 <- xlsx::CellStyle(wb) +
  xlsx::DataFormat("0.000000") +
  xlsx::Alignment(wrapText = TRUE, horizontal = "ALIGN_RIGHT") +
  xlsx::Font(wb, name = "Arial", heightInPoints = 9)

cs10_18 <- xlsx::CellStyle(wb) +
  xlsx::DataFormat("0.00") +
  xlsx::Alignment(wrapText = TRUE, horizontal = "ALIGN_RIGHT") +
  xlsx::Font(wb, name = "Arial", heightInPoints = 9)

format0 <- rep(list(cs0),18)
format1 <- rep(list(cs1), 1)
format2_6 <- rep(list(cs2_6), 5)
format7_8 <- rep(list(cs7_8), 2)
format9 <- rep(list(cs9), 1)
format10_18 <- rep(list(cs10_18), 9)

format <- c(format1, format2_6, format7_8, format9, format10_18)
names(format) <- seq(1, dim(df)[2], by = 1)
names(format0) <- seq(1, dim(df)[2], by = 1)


#Add data and apply the created style
# write dataframe to replace "Data" sheet row 10: Publish date
format_pub <- c(list(cs0), rep(list(cs1), 17))
names(format_pub) <- seq(1,18)

xlsx::addDataFrame(
```

```
  rpub,
  sheet,
  col.names = FALSE,
  row.names = FALSE,
  startRow = 10,
  startColumn = 1,
  colStyle = format_pub
)


xlsx::addDataFrame(
    df,
    sheet,
    col.names = FALSE,
    row.names = FALSE,
    startRow = row.to.change,
    startColumn = 1,
    colStyle = format
)

xlsx::addDataFrame(
  dfTRI,
  sheet,
  col.names = FALSE,
  row.names = FALSE,
  startRow = row.to.change + 1,
  startColumn = 1,
  colStyle = format
)

xlsx::saveWorkbook(wb, file = "<INTERNAL_PATH>/f01d.xlsx")
```

```
# Set JAVA HOME (required due to the use of the `xlsx` package)
Sys.setenv(JAVA_HOME="C:\\Program Files\\Wolfram
Research\\Mathematica\\13.2\\SystemFiles\\Java\\Windows-x86-64")

#' transform_data
#'
#' @param excel_data A list of Excel data.frames loaded via
common$import_excel(blueprint_file)
#' @param sql_data A list of SQL data.frames loaded via
common$import_sql(blueprint_file)
#'
#' @return A list of transformed object ready to be exported to
Excel
transform_data <- function(excel_data, sql_data, blueprint_file,
debug = FALSE) {
  ## Load blueprint (to get final output column names in
abbreviation)
  blueprint <- read_yaml(blueprint_file)
  final_col_names <- c("Series ID",
unlist(blueprint$output$template.xlsx))

  ## Utility Functions
  # Round function
  round2 = function(x, n) {
    z = abs(x)*10^n
    z = z + 0.5
    z = trunc(z)
    z = z/10^n
    z
  }

  # start of the month function
  som <- function(x) {
    as.Date(format(x, "%Y-%m-01"))
  }

  #Last day of previous month
  LastDay <- som(Sys.Date())-1

  # f1 revision utility
  fun_revision <- function(query_data = SQL_dailydata) {
    # Convert to double for case_when - mutate new columns for
changes
    revisions_f1 <- query_data %>%
      mutate(numberoftransactions =
as.double(numberoftransactions),
             Counterparties = as.double(Counterparties))

    # Calculate monthly mean/sum for withheld data
    revisions_f1hist <- revisions_f1 %>%
```

```
      group_by(month = ceiling_date(date, "month",
change_on_boundary = T) %m-% days(1)) %>%
      mutate(monthly_highest_IBOC = round2(mean(maximumrate), 2),
             monthly_lowest_IBOC = round2(mean(minimumrate), 2),
             monthly_volume = round2(sum(totalvolume/1000000),
0),
             monthy_transactions = sum(numberoftransactions))

    # Export CSV for checking
    # write.csv(revisions_f1hist, 'Table
F1/f01hist_revisions.csv')

    revisions_f1hist <- revisions_f1hist %>%
      select(month, monthly_highest_IBOC, monthly_lowest_IBOC,
monthly_volume, monthy_transactions) %>%
      distinct()

    return(revisions_f1hist)
  }

  fun_trans <- function(f01, f01hist, revisions_f1hist,
                        LastDay) {

    if (as.Date(f01hist[[nrow(f01hist), "Title"]],
                tryformat = c("%d-%b-%Y", "%Y-%m-%d")) != LastDay
| debug){

      # Get data from previous month
      f01 <- subset(f01, as.Date(Title) > som(
        som(Sys.Date()) - 1)-1 & as.Date(Title) <
som(Sys.Date()))
      f01_new <- revisions_f1hist %>% filter(month == LastDay)

      # Divide the set into three part for different calculation
      # mean
      columns_part1 <- c(2,4) # columns_part1 <- c(2,4,5, 6)
      #revised mean
      columns_part2 <- c(2, 3, 4, 5)
      columns_part3 <- c(10,11,12,13,14,15,16,17,18)

      f01_temp_part1 <- f01[,columns_part1]
      f01_temp_part2 <- f01_new[,columns_part2]
      f01_temp_part3 <- f01[,columns_part3]

      # Preprocess
      f01_temp_part1 <- f01_temp_part1[sapply(f01_temp_part1,
is.numeric)]
      f01_temp_part3 <- f01_temp_part3[sapply(f01_temp_part3,
is.numeric)]
```

```
      # Part1
      f01_monthly_part1 <- as.data.frame(lapply(f01_temp_part1,
mean, na.rm = T))
      f01_monthly_part1 <- round2(f01_monthly_part1,2)

      # Part2
      f01_monthly_part2 <- f01_temp_part2
      format(round(f01_monthly_part2,0),nsmall=0)

      # Part3
      f01_monthly_part3 <- as.data.frame(lapply(f01_temp_part3,
mean, na.rm = T))
      f01_monthly_part3 <- round2(f01_monthly_part3, 2)

      # Union historical data with the one new row from latest
month
      df_f01hist <- bind_rows(
        f01hist,
        setNames(
          bind_cols(Title = LastDay,
                    f01_monthly_part1,
                    f01_monthly_part2,
                    f01_monthly_part3),
          names(f01hist))
      )

      return(
        list(Data = setNames(df_f01hist, final_col_names))
      )
    } else {
      message(paste0('<i class="fa-solid fa-circle-exclamation"
style="color: #f07400;"></i> No new data added. Last row in
f01hist has the date ',
                     tail(f01hist$Date,1),
                     ", which is up to date (i.e. at the end of
last month)."))
      return(
        list(Data = setNames(f01hist, final_col_names)) # Use
historical values from the Prod folder
      )
    }
  }

  message(paste0(Sys.time(), " Utility functions loaded!"))

  ## Data Transformation
  revisions_f1hist <- fun_revision(sql_data$SQL_dailydata)

  f01hist <- excel_data$f01d_hist %>% mutate(Title =
as.Date(Title)) # Convert time stamp to date
```

```
  f01 <- excel_data$f01d %>% mutate(Title = as.Date(Title)) #
Convert time stamp to date

  result <- fun_trans(f01, f01hist, revisions_f1hist, LastDay)

  message(paste0(Sys.time(), " Data transformation completed!"))

  ## Return output
  return(result)
}
```