MINISTRY OF SOCIAL DEVELOPMENT
TE MANATŪ WHAKAHIATO ORA

Youth Health and Wellbeing Survey 2025 - Analysis Methodology

Contents

Purpose

This document has been created to complement the Fieldwork technical report. The Fieldwork technical report provides information about the survey from design and sample selection, through to data collection and weighting of the data to represent the national population of secondary school students. This Analysis technical summary explains how the data was used to produce the results found in the Overview infographic report and Data tables.

Confidentiality

For any data or reporting entering the public domain there is a responsibility to Youth Health Wellbeing Survey (YHWS) participants to keep their personal information safe. This means applying aggregation and suppression rules to findings where counts of respondents who selected certain answers is low.

The Stats NZ Output Guide[1] has been used throughout this analysis as it must be used to inform how confidentiality rules are applied to survey data. In addition, the following points should be followed;

- For unweighted or weighted counts, where the unweighted count is less than six, it should be supressed.
- After suppression, all unweighted counts should be randomly rounded to a base of three.
- After suppression, all weighted counts should be rounded to a base of 100.
- To ensure confidentiality of answers by a school, results should only be created when at least three schools (entities) contributed to the counts. If fewer than three schools contributed to the counts, the count should be supressed.

In cases where suppression applies, the category should either be aggregated with another category, or omitted from tables and charts in a way that does not allow the omission to be unwound by reverse calculating off other data contained in the published version.

Missing data

A key interest for this data was in comparisons over time. To produce insights that are more consistent with those produced in the previous iteration of reporting, the 2021 What About Me? (WAM) survey, results have been produced using a similar approach. WAM analysis reported percentages based on people who answered the question “validly” that is by providing an answer and providing an answer that was not just “not sure” or “prefer not to say” - this approach has been followed for this analysis too. This approach is supported by partner agencies, the Ministries of Health and

[1] https://www.stats.govt.nz/assets/Methods/Microdata-Output-Guide-2020-v5-Sept22update.pdf

Justice, who contributed to the costs of delivering the YHWS 2025 and will be reporting on data from it.

What constitutes missing data

Questions could be missing data for a range of reasons:

- A student stopped the survey part way through: young people could choose to stop answering questions at any point so answers could be missing because the respondent stopped the survey before completing the survey. However, 98.6% of people who started the survey fully completed it, so this is a small contributor.
- Question logic: after piloting the survey question/skip logic was introduced to some questions to avoid asking irrelevant or unnecessary questions. For example, only people who said they had been bullied were asked about the kind of bullying they experienced.
- A student choosing "prefer not to say": As required by the Health and Disability Ethics Committee (HDEC) in the ethics review process, almost all questions in the survey were optional. Therefore, young people could say "prefer not to say" to move past a question they did not want to answer.
- Many questions also included a "not sure" option: to take a consistent approach to kinds of missing data, "not sure" has been included as a missing value except for two cases:
  1. "Who are you attracted to (how do you experience attraction)?" where being "not sure" or questioning is a meaningful answer that fed into the definition for Rainbow youth, and
  2. "Do you currently experience any mental health conditions?" as people who were "not sure" would be asked follow-up questions about the kind of condition as they may have one but may be unsure about experiencing symptoms currently.

How missingness is reported

For questions that were asked of anyone (without any skip/question logic): the proportion of missing values are calculated as the number of people whose answer was missing or said "prefer not to say" or "not sure" out of the full count of participants.

For questions that were only asked of a subset of people (based on skip/question logic): the proportion of missing values are based on the number of people whose answer was missing or said "prefer not to say" or "not sure" out of the people who should have been presented the question, as based on question logic.

Where missing values were too small (fewer than six people or based on results from less than three schools/kura) they have not been reported, as outlined in the confidentiality section.

As noted in the Fieldwork technical report, following the pilot for the YHWS 2025 actions were taken to minimise missing data such as:

- Removing questions with non-response exceeding 50%.
- Changing the wording in some questions for clarity.
- Applying skip logic to more questions to avoid asking unnecessary and irrelevant questions.

However, some questions still produced high levels of missing answers. As in the 2021 WAM[2] survey, we expect some key reasons a higher number of respondents skipped or selected "not sure" to specific questions could include:

- Respondents not knowing how they felt about a particular question.
- Respondents not understanding what the question was asking them.
- Some respondents may have skipped questions they did not see as relevant to them or had many options they did not want to read.
- Survey fatigue may have led to participants not answering questions with a large number of options.

The Fieldwork technical report details questions with the highest non-response rate. In future questionnaire design, MSD will consider the wording or inclusion of questions with variables that have particularly high missing, "not sure", or "prefer not to say" proportions.

Estimating proportions and weighted totals

Estimates were based upon calibrated weights. Generating weights in a sample survey is essential to ensure that results accurately represent the target population. Participants in the sample had weights calculated based on their characteristics (Regional area, Gender, Ethnicity, and Schooling Equity Index (EQI) quartile), for more information about how these were created please see the Fieldwork technical report[3].

Weighting compensates for potential biases arising from unequal selection probabilities, non-response, and discrepancies between the sample and known population characteristics. These weights are used to provide estimates intended to more closely match those of the New Zealand secondary school population[4].

Most statistics in the YHWS 2025 data tables are rounded weighted totals or proportions based on weighted data. That is, survey estimates of:

[2] www.msd.govt.nz/documents/about-msd-and-our-work/publications-resources/consultations/youth-health-and-wellbeing-survey-results/the-national-youth-health-and-wellbeing-survey-2021-overview-report-september-2022.pdf

[3] www.msd.govt.nz/about-msd-and-our-work/publications-resources/consultations/youth-health-and-wellbeing-survey-results/index.html

[4] More specifically years 9–13 at secondary schools, kura kaupapa Māori, and composite schools.

- the estimated rounded weighted total number of people with a particular characteristic, or who said a particular response, and
- the percentage (based on weighted data) of people with a particular characteristic, or who answered a question with a particular answer.

A description of these types of statistics follows.

Calculating weighted totals

Estimates of weighted totals were calculated as the sum of weights for the respondents who had a particular variable of interest. For example, the estimate of the weighted total number of people who strongly agreed to the statement "I feel a sense of belonging to Aotearoa/New Zealand as a whole" would be given by the sum of weights for all people who answered that they strongly agreed.

Calculating percentages based on weighted data

Estimates[5] for the population

The proportion of the population who belong to a particular group (such as the proportion of the population who strongly agreed to the statement "I feel a sense of belonging to Aotearoa/New Zealand as a whole") was estimated by calculating the sum of the weights of the respondents in the group divided by the sum of the weights of all respondents whose response was not missing. E.g.

$$Estimate = \frac{\text{Sum of weights for people who strongly agree}}{\text{Sum of weights for people who strongly agree, agree, are neutral, disagree, or strongly disagree}}$$

Estimates for a subgroup

The proportion of people in a population group who belong to a subgroup (such as the proportion of the population who strongly agreed to the statement "I feel a sense of belonging to Aotearoa/New Zealand as a whole", who are female) was estimated by calculating the sum of the weights of the respondents in the subgroup (people who strongly agreed to the statement "I feel a sense of belonging to Aotearoa/New Zealand as a whole" and are female) divided by the sum of the weights of the respondents in the population group whose response was not missing (people who are female and did not provide a missing response).

Estimates for multichoice questions

Estimates for multichoice questions such as "Which languages can you have an everyday conversation in?" were based on the proportion of people who answered each option ("English", "Te Reo Māori", "New Zealand Sign Language" or "Another")

[5] Proportions will also be referred to as estimates.

out of the people who answered with at least one "valid" response. This means people who were missing from this question or who only answered with "prefer not to say" are excluded.[6] This is consistent with approaches outlined above as these are effectively missing responses.

Measures of error

Standard errors

Sampling error arises from the survey being taken from a subset of the population rather than using the whole population. To estimate how the results could have been different if some of the respondents had not been a part of the survey, replicate weights were produced. Please see the Fieldwork technical report for information about how the replicate weights were produced.

Replicate weights are used to calculate standard errors for estimates derived from YHWS 2025 data. Replicate weights were calculated using the delete-a-group jackknife method to accommodate the sample design and weighting for the YHWS 2025.

The delete-a-group jackknife method, like other resampling methods, uses the variation between the results for many sample "replicates" to estimate sampling variances.

Standard errors (SE) can be found by:

$$se = \sqrt{\frac{R-1}{R}\sum_{r=1}^{R}(estimate - \ estimate_{(j)})^2}$$

Where:

- The $Estimate$ is the proportion of the population who belong to a particular group as specified in the Calculating percentages based on weighted data.
- R is the number of replicate weights.
- $Estimate_{(j)}$ is the estimate produced using the replicate weights where the jth group was deleted (using replicate weights of set j).

Generally, the smaller the pool of respondents that answered the question, the higher the margin of error estimates.

Calculating confidence intervals

Uncertainty in the survey results can be quantified by using 95% confidence intervals. The upper and lower bounds of this confidence interval describe the range of the

[6] Other multichoice questions would also not include anyone who only answered "not sure", however, this was not an option for this question.

estimates that would be obtained 95% of the time if the survey had been run multiple times. Confidence intervals can be calculated using the normal approximation method. The upper and lower limits of the 95 percent confidence interval were found by:

$$Confidence\ interval = estimate\ \pm 1.96 * \text{se}$$

As in Stats NZ publications such as GSS[7], in the data tables $1.96 * \text{se}$ is provided as the Absolute Sampling Error and can be used to find the confidence interval.

Statistical significance

A standard approach is to regard the difference between two estimates as significant if their confidence intervals do not overlap. Results should not be considered different if their confidence intervals overlap, even where the point estimates appear to be far apart.

Within this survey, utilising other statistical methods (such as chi-squared tests) may result in differences in which findings were identified as statistically significant or not, particularly in fringe cases.

When comparing to other surveys in the series differences that appear significant could have been impacted by differences such as wording changes, survey methodology, or different contexts – such as a question being asked during a significant event such as COVID-19. Because of this, drawing conclusions about differences from just one survey to another should be viewed with caution. It is highly recommended instead to compare across more than one survey to understand potential trends over time.

Relative Sampling Error

Estimates with a Relative Sampling Error (RSE) between 30 and 50 percent are flagged with *, and need to be viewed with caution. Estimates with relative sampling error over 50% are flagged with ** and should be considered unreliable for most purposes. Calculated as:

$$\text{RSE} = \frac{estimate}{\text{se}}$$

[7] wellbeing-statistics-2023-updated-26-march-2025.xlsx

Appendix 1 - data processing specific to this survey

Processing for the overall population

- As WGSS_Disability_Score was in the data dictionary with no restrictions for missingness, where all of the WGSS questions were “prefer not to say” this derived variable was changed to missing/null.
- As Rainbow was in the data dictionary with no restrictions for missingness, values were changed to missing/null where a person said “prefer not to say” for all questions that went into it: birth_sex, gend, and attraction.
- If Parent_Birth_Status = “Unknown /Refused” then treated as null/missing.
- To properly report on counts of junior and senior, the “other” were merged with missing/null for the Junior_Senior variable.
- Derived variables for total ethnicities were restricted to people who provided an ethnicity and did not just say “prefer not to say”.
- Similar to how the multichoice question answers are processed, Paid_Employment was changed to null/missing where no valid answer was given to the relevant multichoice question (a valid answer would be choosing one of Work_PartTime, Work_FullTime, Work_Holidays, Work_School or Work_No, e.g. those that did not non-respond by choosing only “not sure”, “prefer not to say” or were missing as had stopped responding).

Processing in breakdowns

Many responses were grouped to minimise the need for suppression due to low counts, e.g. “Strongly agree” being grouped in with “Agree”.

For the variable HOME_BEDROOMS, where they answered that they had 0 bedrooms in their home (“None / Does not apply to me”), these were grouped with null/missing values as this answer seemed a better fit than grouping with another count of bedrooms.

The variables CONTRACEPTION_OTHER and CONTRACEPTION_NONE are excluded from the output of all breakdowns due to low counts.

Where the number of people answering was too low for some of the options within the variables FAITH or BIRTH_COUNTRY, they have been grouped into “Other” rather than being removed.

Males

- Too few had people indicated birth sex of “intersex”, so these have been grouped with people who said birth sex was female to meet confidentiality rules.
- Menstruation questions will not be provided for this group as the count of responses is too low.

Females

- Too few had people indicated birth sex of “intersex”, so these have been grouped with people who said birth sex was male to meet confidentiality rules.
- The variable LIVE_WORKMATES was excluded due to a low number of responses.

Rainbow

- In the variable EDU_YEAR, people who responded with “Other” were grouped with missing values to meet confidentiality rules.
- The variables LIVE_ALONE, LIVE_WORKMATES, CONTRACEPTION_IUD, SUBST_FX_SNUS, and SUBST_FX_SYNTH were excluded due to a low number of responses.

Auckland

- In the variable EDU_YEAR, people who responded with “Other” were grouped with missing values to meet confidentiality rules.
- The variables HEALTHCARE_UNF_SEX, LIVE_WORKMATES, SUBST_FX_SYNTH, and URBAN_RURAL_INDICATOR were not reported due to a low number of responses in certain categories.

Rest of North Island

- The variable LIVE_WORKMATES was excluded due to a low number of responses.

South island

- Too few had people indicated birth sex of “intersex”, so these have been changed to missing responses to meet confidentiality rules.
- The variables LIVE_WORKMATES, CONTRACEPTION_DEPO, and OTHER_TOTAL_ETH were excluded due to a low number of responses.

Junior

- Too few had people indicated birth sex of “intersex”, so these have been changed to missing responses to meet confidentiality rules.
- The variables LIVE_WORKMATES, EDU_YEAR, AGE, CONTRACEPTION_DEPO, CONTRACEPTION_IUD, and SUBST_AGE_SYNTH were not reported due to a low number of responses in certain categories.

Senior

- The variables LIVE_WORKMATES, EDU_YEAR, and AGE were not reported due to a low number of responses in certain categories.

Māori

- The variables LIVE_WORKMATES and OTHER_TOTAL_ETH were not reported due to a low number of responses.

Pacific

- In the variable EDU_YEAR, people who responded with “Other” were grouped with missing values to meet confidentiality rules.
- HOME_BEDROOMS has been further grouped (1-2 bedrooms) to avoid suppression.
- The variables CONTRACEPTION_DEPO, CONTRACEPTION_IUD, HEALTHCARE_UNF_SEX, LIVE_FLATMATES, LIVE_WORKMATES, and OTHER_TOTAL_ETH were not reported due to a low number of responses.

Asian

- Too few had people indicated birth sex of “intersex”, so these have been changed to missing responses to meet confidentiality rules.
- In the variable EDU_YEAR, people who responded with “Other” were grouped with missing values to meet confidentiality rules.
- HOME_PEOPLE has been further grouped (1-2 people) to avoid suppression.
- The variables CONTRACEPTION_DEPO, CONTRACEPTION_IUD, EDU_TYPE_ALT_ED, FUT_DO_NOTHING, HEALTHCARE_UNF_GEND, HEALTHCARE_UNF_SEX, LIVE_ALONE, LIVE_PARTNERSPARENTS, LIVE_WORKMATES, OTHER_TOTAL_ETH, SLEEP_BOARDING_EMERG, SLEEP_HOSTEL, SUBST_FX_CIGARETTE, SUBST_FX_GLUE, SUBST_FX_MEDS, SUBST_FX_OTHER, SUBST_FX_SNUS, SUBST_FX_VAPE, SUBST_AGE_OTHER, SUBST_AGE_SYNTH were not reported due to a low number of responses in certain categories.

European

- The variable LIVE_WORKMATES was not reported due to a low number of responses.

Non-rainbow

- No alterations needed.

WGSS disability

- The variable LIVE_WORKMATES was not reported due to a low number of responses.

Non-disability

- The variable LIVE_WORKMATES was not reported due to a low number of responses.

Non-Māori

- The variables LIVE_WORKMATES and KNOW_IWI were not reported due to a low number of responses.

Non-Pacific

No additional changes than the general approach listed (e.g. grouping of BIRTH_COUNTRY or FAITH).

Non-Asian

No additional changes than the general approach listed (e.g. grouping of BIRTH_COUNTRY or FAITH).

Non-European

- In the variable EDU_YEAR, people who responded with “Other” were grouped with missing values to meet confidentiality rules.
- The variable CONTRACEPTION_IUD were not reported due to a low number of responses.

Appendix 2 - Code used to produce estimates and error

Snowflake SQL code used for analysis

```
-- produce survey proportions/rates
create or replace procedure get_survey_prop(
    "in_table" varchar,
    "out_table" varchar,
    "final_weight" varchar,
    "replicate_weight" varchar,
    "by_column" varchar,
    "filter" varchar default ' ',
    "provider_id" varchar default ' ',
    "is_temporary" boolean default true,
    "nrep" numeric default 100,
    "std_devs" numeric(30,10) default 1.96
    )
returns varchar
language sql
execute as caller
as
declare
    sqlstmt string default ' ';
    variable_headers string default ' ';
    sestmt string default ' ';
    summation_totals string default ' ';
    summation_by_columns string default ' ';
```

```
    entity_count_stmt string default ' ';
    entity_count_call string default ' ';
    counter2 integer default 1;
    counter1 integer default 1;
    current_col string;
    col_list string;
    part_count integer;
    check_existence string;
    by_columngrp string;
    prop string;
    does_col_exist boolean;
begin

/*
Title: produce rate estimates from survey data along with estimates of error.

Purpose: uses Stats NZ jack-knife approach to creation of rates, SE and RSE
estimates. This is the process used for creation of survey summaries such as the
General Social Survey.

Inputs:
    in_table (string): input table name - it does not need to be a fully qualified table
name.
    out_table (string): output table name.
    final_weight (string): name of the final weights in the data.
    replicate_weight (string): name of the replicate weight, this should be the text that
preceeds a number.
    by_column (string): variables that you want to breakdown by, there can be
multiple by_column separated by a space.
    filter (string): optional, to add a filter, include where in the string if providing e.g.
'where age = 5'.
    provider_id (string): optional, to provide an entity_count to understand the number
of distinct entities contributing to an answer.
    is_temporary (boolean): optional, decide whether the output table is temporary,
default is TRUE.
    nrep (numeric): optional, number of replicate weights, values from 1-nrep will be
appended to the replicate_weight values teh default is 100.
    std_devs (numeric(30,10)): the standard deviations requested for the calculation,
default is 1.96.

Notes:
- This stored procedure assumes that each line of input data relates to a different
survey respondent, input data have been prepared consistent with the use of the
jackknife estimator and at least one by variable is specified (if wishing to create the
total you may create a column only populated with 'Total').

- The Youth Health and Wellbeing Survey (YHWS) data requires suppression of results
when fewer than three educational providers contribued to results. This is a key
```

example of when you would want to use provider_id to get a count of distinct provider_ids by grouping.

- The output table is temporary by default but can be made permanent by setting the is_temporary parameter to FALSE.

Output: a table that has the estimate of the rate and Stats NZ SE and RSE, broken down by the by_column.

- variable_1 to variable_n: The variables passed to the procedure through by_column will be displayed in the order provided
- variable_level_1 to variables_level_n: the values applicable from the relevant variable.
- raw_count_individuals: the raw count of individuals that made up the proportion of that paricular breakdown.
- weighted_count_individuals: the weighted count of individuals from the weight variable.
- (if applicable) entity count: an entity count based on the count of distinct entities from the provider_id input.
- est: the proportion of people from the total breakdown sum(final weight for by_column combination)/sum(final weight).
- se: the sampling error around the est value, found by sqrt(((nrep-1)/nrep)*(suqare(est - est based on replicate_weight1) + ... + suqare(est - est based on replicate_weight1)))
- snz_ase: the absolute standard error as in published GSS statistics and used to calculate confidence intervals (est + snz_ase & est - snz_ase), calculated as 1.96*se.
- rse: The relative sampling error around the est value, calculated from the replicate weights. Expressed as a percentage of est. Calculated as - (se/est) – this can be multiplied by 100.

Example:

```
call get_survey_prop(
    in_table =>  'yhws_data',
    out_table =>  'yhws_data_prop',
    final_weight =>  'weight',
    replicate_weight =>  'weight_rep',
    by_column =>  'gend',
    filter =>  'where age = 5',
    provider_id => 'provider_id',
    is_temporary => true,
    nrep => 30,
    std_devs => 1.96
    );

*/

    -- process the columns and return a clean list
    col_list := clean_column_list(by_column);
```

```
    -- process and make sure strings are clean
    final_weight := clean_column_list(final_weight);
    replicate_weight := clean_column_list(replicate_weight);
    in_table := clean_column_list(in_table);
    out_table := clean_column_list(out_table);
    provider_id := clean_column_list(provider_id);

    -- check the where clause is in the filter
    if(clean_column_list(filter) != '') then
        if(filter not ilike '%where%' ) then
        return '1 - Failed to run, filter needs to be removed or look like: where age = 5';
        end if;
    end if;

    -- check that the final_weight exists
    call column_exists(:in_table, :final_weight) into :does_col_exist;
    if (does_col_exist = false) then
        return '1 - Failed to run: final_weight column does not exist';
    end if;

    -- check that the replicate_weight exists
    call column_exists(:in_table, :replicate_weight || :nrep) into :does_col_exist;
    if (does_col_exist = false) then
        return '1 - Failed to run: replicate_weight column to specified nrep does not
exist';
    end if;

    -- check that the replicate_weight exists
    call column_exists(:in_table, :provider_id) into :does_col_exist;
    if (does_col_exist = false and provider_id != '') then
        return '1 - Failed to run: provider_id column does not exist';
    end if;

    -- build group columns string for group statements and joining conditions
    select count(*) into part_count
    from table(flatten(input => split(:col_list, ' ')));
    counter1 := 1;

    while (counter1 <= part_count) do
        current_col := split_part(:col_list, ' ', counter1);

         -- check the columns exist
        call column_exists(:in_table, :current_col) into :does_col_exist;
        if (does_col_exist = false) then
            return '1 - Failed to run: not all columns requested exist';
        end if;

        if (counter1 = 1) then
```

```
            by_columngrp := current_col;
            variable_headers := ' '''||current_col||''' as variable_' || counter1 || ', ' ||
current_col || ' as variable_' || counter1 || '_level';
        else
            by_columngrp := by_columngrp || ', ' || current_col;
            variable_headers := variable_headers || ', '''||current_col||''' as variable_' ||
counter1 || ', ' || current_col || ' as variable_' || counter1 || '_level';
        end if;

        counter1 := counter1 + 1;

    end while;

    -- if a provider_id is given, build an entity count
    if (provider_id = ' ' or provider_id = '') then
        entity_count_stmt := ' ';
        entity_count_call := ' ';
        else
        entity_count_stmt := ', count(distinct ' || provider_id || ') as entity_count ';
        entity_count_call := ' entity_count, ';
    end if;

    -- build summation and se calculation
    summation_totals := 'sum(' || final_weight || ') as final_weight_total, ';
    summation_by_columns := 'sum(' || final_weight || ') as final_weight, ';
    prop := 'final_weight/final_weight_total*100 as est, ';

    while (counter2 <= nrep) do
        summation_totals := summation_totals || 'sum(' || replicate_weight || counter2
|| ') as final_weight_' || counter2 || '_total';
        summation_by_columns := summation_by_columns || 'sum(' ||
replicate_weight || counter2 || ') as final_weight_' || counter2;
        prop := prop || 'final_weight_' || counter2 || '/final_weight_' || counter2 ||
'_total*100 as est_' || counter2;
        sestmt := sestmt || 'square(est - est_' || counter2 ||')';
        if (counter2 < nrep) then
            summation_totals := summation_totals || ', ';
            summation_by_columns := summation_by_columns || ', ';
            prop := prop || ', ';
            sestmt := sestmt || ' + ';
        end if;
        counter2 := counter2 + 1;
    end while;

    -- sql statement
    sqlstmt := -- find the total weights across the by_columns
            'with weight_totals_grouped as (
                select ' || by_columngrp || ', ' || summation_by_columns ||
                ', count(*) as raw_count_individuals ' ||
```

```
                ', sum(' || final_weight || ') as weighted_count_individuals ' ||
                entity_count_stmt ||
                ' from ' || in_table || ' ' || filter || ' group by ' || by_columngrp || '), ' ||
            -- find the total weights for each replicate
            'weight_totals as (
                select ' || summation_totals ||
                ' from ' || in_table || ' ' || filter || '), ' ||
            -- join so the table has both totals for the weighted data and totals for the
by_columns
            'joined_data as (
                select b.* , a.*
                from weight_totals as a
                    cross join weight_totals_grouped as b), ' ||
            -- summarise and produce error estimates
            'summary as (select ' || by_columngrp || ',
                raw_count_individuals,
                weighted_count_individuals, ' ||
                entity_count_call  ||
                prop || ', ' ||
                'sqrt((' || nrep || '-1)/' || nrep || ')*(' || sestmt || ')) as se, '  ||
                std_devs ||'*se as snz_ase, '  ||
                'iff(est != 0,  se/est, null) as rse ' ||
                'from joined_data) ' ||
            -- final table for output
            'select ' ||  variable_headers || ',
                raw_count_individuals,
                weighted_count_individuals, ' ||
                entity_count_call  ||' est, se, snz_ase, rse from summary order by ' ||
by_columngrp
            ;

    -- execute immediate sqlstmt;
    execute immediate 'create or replace ' || iff(is_temporary, 'temporary', ' ') || ' table
' || out_table || ' as ' || sqlstmt;

    -- return statement based on whether it was completed or not
    return '0 - completed: ' || 'create or replace ' || iff(is_temporary, 'temporary', ' ') ||
' table ' || out_table || ' as ' || sqlstmt ;
    exception
        when other then
            return '1 - Failed to run: ' || sqlstmt;
end;
```

Supporting functions required

```
/*
TITLE: COLUMN_EXISTS
```

```
PURPOSE: Function that returns true or false based on whether column exists in
in_table

INPUTS:
    IN_TABLE: name of input table
    IN_COLUMN: name of date or datetime column

OUPUT: Boolean
    true if in_column exists in in_table
    false if in_column does not exist

NOTES:
    Function to identify if a column exists in a table. The column name is not
    case senstive.

EXAMPLE 1:
call column_exists(
    in_table => 'clients'
    ,in_column => 'service_code'
    );


EXAMPLE 2: Within a stored procedure
declare
    svc_col_exists boolean;
begin
    call column_exists(:in_table,'SERVICE_CODE') into :svc_col_exists;

*/


create or replace temporary procedure column_exists(
    in_table varchar
    ,in_column varchar
)
returns boolean
language sql
execute as caller
as
$$
declare
    rs              resultset;
    table_name      varchar;
    schema_name     varchar;
    column_name     varchar;
begin
    rs := (execute immediate 'show columns in table ' || in_table);
    let cur cursor for rs;
    open cur;
```

```
    loop
        fetch cur into table_name, schema_name, column_name;

        if(column_name is null) then
            break;
        else
            if(upper(column_name) = upper(:in_column)) then
                return true;
            end if;
        end if;

    end loop;
    close cur;

    return false;
end;
$$;



/*
TITLE: Clean column list string

PURPOSE: Clean and standardise a string of column names for later processing

INPUT: col_list: a string containing a list of column names

OUPUT: a string with cleaned and standardised column names.

NOTES:
Cleaning and standardised involves:
    - Replace every character that is NOT valid as an unquoted Snowflake
    identifier (A-Z, 0-9, _ , $) with one space
    - Collapse any multi-whitespace characters to one space (handles tabs/new-lines)
    - Trim leading / trailing blanks
    - Upper-case the result for case-insensitive matching

EXAMPLE:
  ID_COLUMNS:=CLEAN_COLUMN_LIST(ID_COLUMNS);
*/

CREATE OR REPLACE temporary FUNCTION CLEAN_COLUMN_LIST(col_list STRING)
RETURNS STRING
IMMUTABLE   -- deterministic
AS
$$
UPPER(
    TRIM(
        REGEXP_REPLACE(
```

```
        REGEXP_REPLACE(col_list, '[^A-Za-z0-9_$]+', ' '),
        '\\s+', ' '
      )
    )
  )
$$;
```