Work Instructions

Number: **005**

Name: **pMA Profiling**

Title: Profiling of datasets using phpMyAdmin

Status: Approved

Version: 1.0

Date: 2023-06-30

Authors: Michael Rubinigg

This document is intended for the regulation of processes within the EU Pollinator Hub. It is published under the CC BY 4.0 licence. You may distribute, transform, adapt and build upon the material in any medium or format, including for commercial purposes, as long as you give credit to the author.

Table of contents

1 Document history 3
1.1 Current Release 3
1.2 Change history 3
2 Abbreviations 4
3 Introduction 5
4 Validity 6
4.1 Area 6
4.2 Time 6
4.3 Related Documents 6
5 Procedures 7
5.1 Setup and configuration 7
5.2 Perform database queries 8
5.3 Structural discovery 9
6 References 15
7 Annexes 16
7.1 SQL queries 16

1 Document history

1.1 Current Release

1.1.1 Drafted

Submitted on 2023-06-05 by Michael Rubinigg, quality manager.

1.1.2 Reviewed

Reviewed on 2023-06-30 by Noa Simòn Delso, general manager.

1.1.3 Approved

Approved on 2023-06-30 by Michael Rubinigg, quality manager.

1.2 Change history

n.a.

2 Abbreviations

EU	European Union
EUPH	EU Pollinator Hub
GUI	Graphical User Interface
n.a.	not available
SOP	Standard Operating Procedure
SQL	Structured Query Language
WI	Work Instruction

3 Introduction

The purpose of the current WI is a detailed description of the procedure to be used for profiling of datasets (SOP 007) using phpMyAdmin. The results of this procedure are intended to be used for drafting a dataset report (SOP-003).

4 Validity

4.1 Area

The of validity of this document is limited to profiling of datasets using phpMyAdmin.

4.2 Time

The period of validity of this WI begins on 2023-06-30 and ends with its revocation.

4.3 Related Documents

Document type	Number	Name
WI	006	pMA Start

5 Procedures

5.1 Setup and configuration

5.1.1 Starting the software

5.1.1.1 XAMPP Control Panel

- Start the application phpMyAdmin as described in WI-006.

5.2 Perform database queries

5.2.1 Selct a database

5.2.1.1 Select a database

- Select the database on which the processes described below should be performed.

5.2.2 Perform simple queries

5.2.2.1 SQL query box

- Select **SQL** to get to the query box from the horizontal menu.
- Enter the query in the SQL query box.
- Select **Go** to submit the query to the server.

5.2.3 Run routines

5.2.3.1 Import a routine from the WI

- Select **SQL** query box from the horizontal menu.
- **Copy** the content of the text file containing the procedure you want to import and **Paste** it into the SQL query box.
- Select **Go** to submit the query to the server.

5.2.3.2 Run routines

- Select **Routines** from the horizontal menu.
- Select **Execute** for the routine that you want to run.
- Follow the instructions issued by the procedure.

5.3 Structural discovery

5.3.1 Structure

Procedure `STRUCTURE` returns information on the structure of a given table (`tbl`) in a database into a generic table of the name `[tbl]_structure` in database `[db]_profile` and a file of the general name `[dir]\[tbl]_structure_[YYYY-MM-DD hhmmss].csv`. Target database and table are created if necessary.

5.3.1.1 Input

- name of the database to be analysed: `db (VARCHAR(64) CHARSET utf8)`
- name of the table to be analysed: `tbl (VARCHAR(64) CHARSET utf8)`
- target directory of the output file: `dir (VARCHAR(256) CHARSET utf8)`

5.3.1.2 Output

The output is written to:

- file: `[dir]\[tbl]_structure_[YYYY-MM-DD hhmmss].csv`
- table: `[tbl]_structure` in database `[db]_profile`

It contains the following columns:

- `Id`: Number of the row.
- `Name`: Name of the column obtained from column *COLUMN_NAME* in table *TABLE_SCHEMA* of dataset *information_schema*.
- `Format`: Data format as defined in MariaDB, obtained from column *COLUMN_TYPE* in table *TABLE_SCHEMA* of dataset *information_schema*.
- `Length`: Minimum and Maximum length of characters, measured in code points.
- `IsNull`: Field may contain NULL values (`YES`, `NO`), obtained from column *COLUMN_IS_NULLABLE* in table *TABLE_SCHEMA* of dataset *information_schema*.
- `Description`: Content of column, obtained from column *COLUMN_COMMENT* in table *TABLE_SCHEMA* of dataset *information_schema*.

5.3.1.3 Procedure

- Run the query `STRUCTURE` in file *structure.txt* and section 7.1.1.

5.3.2 Content

Procedure `CONTENT` returns information on the content of a given table (`tbl`) in a database into a generic table of the name `[tbl]_content` in database `[db]_profile` and a file of the general name `[dir]\[tbl]_content_[YYYY-MM-DD hhmmss].csv`. Target database and table are created if necessary.

5.3.2.1 Input

- name of the database to be analysed: `db (VARCHAR(64) CHARSET utf8)`
- name of the table to be analysed: `tbl (VARCHAR(64) CHARSET utf8)`
- target directory of the output file: `dir (VARCHAR(255) CHARSET utf8)`

5.3.2.2 Output

The output is written to:

- file: `[dir]\[tbl]_content_[YYYY-MM-DD hhmmss].csv`
- table: `[tbl]_content` in database `[db]_profile`

It contains the following columns:

- `Id`: Number of the row.
- `Name`: Name of the column.
- `Average`: Arithmetic mean of all values in the column.
- `Minimum`: Minimum of all values in the column.
- `Maximum`: Maximum of all values in the column.
- `Ranges`: Range of all values in the column.
- `VoVar`: Coefficient of variation based on the standard deviation of all values in the column.
- `IsNull`: Number and percentage of values in the column containing NULL.
- `IsZero`: Number and percentage of values in the column containing zero.
- `IsBlank`: Number and percentage of values in the column containing blank.
- `IsDistinct`: Number and percentage of values in the column containing blank.

5.3.2.3 Procedure

- Run the query `CONTENT` in file *content.txt* and section 7.1.2.

5.3.3 Duplicates

Procedure `DUPLICATES` returns duplicate records in a given table (`tbl`) in a database into a generic table of the name `[tbl]_duplicates` in database `[db]_profile` and a file of the general name `[dir]\[tbl]_duplicates_[YYYY-MM-DD hhmmss].csv`. Target database and table are created if necessary.

5.3.3.1 Input

- name of the database to be analysed: `db (VARCHAR(64) CHARSET utf8)`
- name of the table to be analysed: `tbl (VARCHAR(64) CHARSET utf8)`
- target directory of the output file: `dir (VARCHAR(255) CHARSET utf8)`

5.3.3.2 Output

The output is written to:

- file: `[dir]\[tbl]_duplicates_[YYYY-MM-DD hhmmss].csv`
- table: `[tbl]_duplicates` in database `[db]_profile`

It contains the following columns:

- `$Id`: Number of the row.
- `$Duplicates`: Number of duplicates of the record.
- Fields of the original table

5.3.3.3 Procedure

- Run the query `DUPLICATES` in file *duplicates.txt* and section 7.1.3.

5.3.4 Profile

Procedure `PROFILE` runs procedures `STRUCTURE` (5.3.1), `CONTENT` (5.3.2) and `DUPLICATES` (5.3.3).

5.3.4.1 Input

- name of the database to be analysed: `db_profile (VARCHAR(64) CHARSET utf8)`
- target directory of the output file: `dir_profile (VARCHAR(255) CHARSET utf8)`

5.3.4.2 Output

See procedures `STRUCTURE` (5.3.1), `CONTENT` (5.3.2) and `DUPLICATES` (5.3.3).

5.3.4.3 Procedure

- Run the query `PROFILE` in file *profile.txt* and section 7.1.4.

5.3.5 Distinct instances overall

Procedure `DISTINCTINSTANCEOVERALL` returns information on distinct data occurrences (case sensitive) in a given column (`col`) in a given table (`tbl`).

5.3.5.1 Input

- table name: `tbl (VARCHAR(64) CHARSET utf8)`
- column name: `col (VARCHAR(64) CHARSET utf8)`

5.3.5.2 Output

- The results are displayed in the execution results of the procedure.

It contains the following fields:

- `Number`: Number of instances.
- `Percent`: Percentage share of instances in total number of rows.
- *`col`*: Name of the column to be analysed

5.3.5.3 Procedure

- Run the query `DISTINCTINSTANCEOVERALL` in file *distinctinstancesoverall.txt* and section 7.1.5

5.3.6 Distinct instances per group

Procedure `DISTINCTINSTANCEGROUP` returns information on distinct data occurrences (case sensitive) in a given column (`colprobe`) per occurrence of another column (`colgroup`) in a given table (`tbl`).

5.3.6.1 Input

- table name: `tbl (VARCHAR(64) CHARSET utf8)`
- column name of the probe: `colprobe (VARCHAR(64) CHARSET utf8)`
- column name of the group: `colgroup (VARCHAR(64) CHARSET utf8)`

5.3.6.2 Output

- The results are displayed in the execution results of the procedure.

It contains the following fields:

- *`colgroup`*: Name of the group
- `Instances`: Number of distinct occurrences per group.
- `I_Percent`: Percentage share of distinct occurrences per group in total number of distinct occurrences.
- `Records`: Number of total records per group.
- `R_Percent`: Percentage share of records per group in total number of records.

5.3.6.3 Procedure

- Run the query `DISTINCTINSTANCEGROUP` in file *distinctinstancesgroup.txt* and section 7.1.6.

6 References

There are no sources in the current document.

7 Annexes

7.1 SQL queries

7.1.1 Structure

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `structure`(IN `db` VARCHAR(64) CHARSET utf8, IN
`tbl` VARCHAR(64) CHARSET utf8, IN `dir` VARCHAR(256) CHARSET utf8)
BEGIN
       SET @cnt_num = 0;
       SET @cnt_tot = 0;

       /* create reporting table*/
    EXECUTE IMMEDIATE CONCAT
       ('CREATE DATABASE IF NOT EXISTS ',db,'_profile');
       EXECUTE IMMEDIATE CONCAT
    ('DROP TABLE IF EXISTS ',db,'_profile.',tbl,'_structure');
       EXECUTE IMMEDIATE CONCAT(
    'CREATE TABLE ',db,'_profile.',tbl,'_structure (
        Id BIGINT NOT NULL AUTO_INCREMENT,
               Name VARCHAR(64) NULL,
        Format VARCHAR(64) NULL,
        Length VARCHAR(256) NULL,
        IsNull VARCHAR(3) NULL,
        Description VARCHAR(1024) NULL,
        PRIMARY KEY (Id)
    ) ENGINE = InnoDB');

       /*get total number of columns*/
    SELECT COUNT(*) INTO @cnt_tot FROM information_schema.COLUMNS WHERE TABLE_SCHEMA=db AND
TABLE_NAME=tbl;

    WHILE @cnt_num < @cnt_tot DO
       SET @cnt_num = @cnt_num+1;

        /* get column name abd data types*/
        SELECT
               COLUMN_NAME, COLUMN_TYPE, IS_NULLABLE, COLUMN_COMMENT
       INTO
               @column_name, @data_type, @is_nullable, @column_comment
        FROM information_schema.COLUMNS
        WHERE TABLE_SCHEMA=db AND TABLE_NAME=tbl AND ORDINAL_POSITION = @cnt_num;

               /* get minimum and maximum length of data*/
       EXECUTE IMMEDIATE
               CONCAT(
                 'SELECT
                              MIN(CHAR_LENGTH(',@column_name,')),
                      MAX(CHAR_LENGTH(',@column_name,'))
                      INTO @lenmin, @lenmax
                      FROM ',db,'.',tbl
                      );

       /*write result into reporting table*/
        EXECUTE IMMEDIATE CONCAT(
               'INSERT INTO ',db,'_profile.',tbl,'_structure (Name, Format, Length, IsNull,
Description)
            VALUES ( @column_name, @data_type, CONCAT(IF(@lenmin=@lenmax,\'\',CONCAT(@lenmin,
\'-\')) , @lenmax), @is_nullable, @column_comment )');

    END WHILE;

       /*export reporting file*/
    SET                                        @file                                          =
CONCAT(REPLACE(dir,'\\','\\\\'),CONCAT(tbl,'_structure_',REPLACE(NOW(),':',''),'.csv'));

       EXECUTE IMMEDIATE
    CONCAT('SELECT * INTO OUTFILE \'',@file,'\'
    FIELDS TERMINATED BY \';\'
    OPTIONALLY ENCLOSED BY \'\"\'
    LINES TERMINATED BY \'\\n\'
    FROM ',db,'_profile.',tbl,'_structure');
```

```
END$$
DELIMITER ;
```

7.1.2 Content

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `content`(IN `db` VARCHAR(64) CHARSET utf8, IN `tbl`
VARCHAR(64) CHARSET utf8, IN `dir` VARCHAR(256) CHARSET utf8)
BEGIN
       SET @cnt_num = 0;
       SET @cnt_tot = 0;

       /* create reporting table*/
       EXECUTE IMMEDIATE CONCAT
       ('CREATE DATABASE IF NOT EXISTS ',db,'_profile');
       EXECUTE IMMEDIATE CONCAT
    ('DROP TABLE IF EXISTS ',db,'_profile.',tbl,'_content');
       EXECUTE IMMEDIATE CONCAT(
    'CREATE TABLE ',db,'_profile.',tbl,'_content (
        Id BIGINT NOT NULL AUTO_INCREMENT,
              Name VARCHAR(64) NULL,
        Average DOUBLE NULL,
        Minimum VARCHAR(256) NULL,
        Maximum VARCHAR(256) NULL,
        Ranges DOUBLE NULL,
        CoVar DOUBLE NULL,
              IsNull VARCHAR(256) NULL,
        IsZero VARCHAR(256) NULL,
        IsBlank VARCHAR(256) NULL,
        IsDistinct VARCHAR(256) NULL,
        PRIMARY KEY (Id)
    ) ENGINE = InnoDB');

       /*get total number of columns*/
       SELECT COUNT(*) INTO @cnt_tot FROM information_schema.COLUMNS WHERE TABLE_SCHEMA=db AND
TABLE_NAME=tbl;

    WHILE @cnt_num < @cnt_tot DO
       SET @cnt_num = @cnt_num+1;

              /* get column name*/
              SELECT COLUMN_NAME INTO @column_name
        FROM information_schema.COLUMNS
        WHERE TABLE_SCHEMA=db AND TABLE_NAME=tbl AND ORDINAL_POSITION = @cnt_num;

              /* get maximum number of digits in a column */
        EXECUTE IMMEDIATE
              CONCAT(
                'SELECT
                              MAX(LENGTH(TRIM(TRAILING  \'0\'  FROM  TRIM(LEADING  \'.\'  FROM
REGEXP_SUBSTR(',@column_name,', \'[.][0-9]*$\')))))
                      INTO @round
                      FROM ',db,'.',tbl
              );

       /* get measures of central tendency and dispersion*/
       EXECUTE IMMEDIATE
              CONCAT(
                'SELECT
                      COUNT(*),
                              IF(MAX(',@column_name,'     REGEXP     \'^[0-9.]*$\')=0,     NULL,
ROUND(AVG(CAST(',@column_name,' AS DOUBLE)), IF(@round<1,1,@round))),
                      IF(MAX(',@column_name,' REGEXP \'^[0-9.]*$\')=0, MIN(',@column_name,'),
MIN(CAST(',@column_name,' AS DOUBLE))),
                      IF(MAX(',@column_name,' REGEXP \'^[0-9.]*$\')=0, MAX(',@column_name,'),
MAX(CAST(',@column_name,' AS DOUBLE))),
                      IF(MAX(',@column_name,'       REGEXP       \'^[0-9.]*$\')=0,       NULL,
ROUND((MAX(CAST(',@column_name,'    AS    DOUBLE))-MIN(CAST(',@column_name,'    AS    DOUBLE))),
IF(@round<1,1,@round))),
                      IF(MAX(',@column_name,'       REGEXP       \'^[0-9.]*$\')=0,       NULL,
STDDEV_POP(CAST(',@column_name,' AS DOUBLE)))
                      INTO @n, @average, @minimum, @maximum, @ranges, @sd
                      FROM ',db,'.',tbl
                      );

       /* get number and percentage of NULL, zero and blank values*/
              EXECUTE IMMEDIATE
                      CONCAT('SELECT COUNT(*) INTO @isnull
```

```
                            FROM ',db,'.',tbl,'
                WHERE ISNULL(',@column_name,')');
                SET @isnullperc = ROUND(((@isnull * 100) /@n),1);
                EXECUTE IMMEDIATE
                       CONCAT('SELECT COUNT(*) INTO @iszero
                            FROM ',db,'.',tbl,'
                 WHERE     !ISNULL(',@column_name,')     AND     ',@column_name,'=\'0\'     AND
LENGTH(TRIM(',@column_name,'))>0');
                SET @iszeroperc = ROUND(((@iszero * 100) /@n),1);
                EXECUTE IMMEDIATE
                       CONCAT('SELECT COUNT(*) INTO @isblank
                 FROM ',db,'.',tbl,'
                            WHERE               !ISNULL(',@column_name,')               AND
LENGTH(TRIM(',@column_name,'))=0');
                SET @isblankperc = ROUND(((@isblank * 100) / @n),1);

                /* get number of distinct occurrences*/
        EXECUTE IMMEDIATE
        CONCAT('SELECT COUNT(DISTINCT(CAST(',@column_name,' AS CHAR CHARACTER SET utf8) COLLATE
utf8_bin)) INTO @isdistinct FROM ',db,'.',tbl);
                SET @isdistinctperc = ROUND(((@isdistinct * 100) /@n),1);
        SET @covar = ROUND(((@sd * 100) / @average),1);

                /*write result into reporting table*/
                EXECUTE IMMEDIATE CONCAT(
            'INSERT  INTO  ',db,'_profile.',tbl,'_content  (Name,  Average,  Minimum,  Maximum,
Ranges, CoVar, IsNull, IsZero, IsBlank, IsDistinct)
            VALUES (@column_name, @average, @minimum, @maximum, @ranges,
            @covar,
            CONCAT(@isnull, \' (\' , @isnullperc, \'%)\'),
            CONCAT(@iszero, \' (\' , @iszeroperc, \'%)\'),
            CONCAT(@isblank, \' (\' , @isblankperc, \'%)\'),
            CONCAT(@isdistinct, \' (\' , @isdistinctperc, \'%)\'))'
        );

    END WHILE;

        /*export reporting file*/
        SET                                     @file                                           =
CONCAT(REPLACE(dir,'\\','\\\\'),CONCAT(tbl,'_content_',REPLACE(NOW(),':',''),'.csv'));

        EXECUTE IMMEDIATE
    CONCAT('SELECT * INTO OUTFILE \'',@file,'\'
    FIELDS TERMINATED BY \';\'
    OPTIONALLY ENCLOSED BY \'\"\'
    LINES TERMINATED BY \'\\n\'
    FROM ',db,'_profile.',tbl,'_content');

END$$
DELIMITER;
```

7.1.3 Duplicates

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `duplicates`(IN `db` VARCHAR(64) CHARSET utf8, IN
`tbl` VARCHAR(64) CHARSET utf8, IN `dir` VARCHAR(256) CHARSET utf8)
BEGIN
       SET @grplst = 0;
    SET @replst = 0;

       /*get list of columns for grouping*/
    SELECT  GROUP_CONCAT(COLUMN_NAME)  INTO  @grplst  FROM  information_schema.COLUMNS  WHERE
TABLE_SCHEMA=db AND TABLE_NAME=tbl;

    /*get list of columns for reporting*/
    SELECT   GROUP_CONCAT(COLUMN_NAME,'   ',COLUMN_TYPE,'   NULL')   INTO   @replst   FROM
information_schema.COLUMNS WHERE TABLE_SCHEMA=db AND TABLE_NAME=tbl;

    /* create reporting table*/
       EXECUTE IMMEDIATE CONCAT
       ('CREATE DATABASE IF NOT EXISTS ',db,'_profile');
       EXECUTE IMMEDIATE CONCAT
    ('DROP TABLE IF EXISTS ',db,'_profile.',tbl,'_duplicates');

       EXECUTE IMMEDIATE CONCAT(
    'CREATE TABLE ',db,'_profile.',tbl,'_duplicates (
        $Id BIGINT NOT NULL AUTO_INCREMENT,
        $Duplicates BIGINT NULL,
               ',@replst,',
        PRIMARY KEY ($Id)
    ) ENGINE = InnoDB');

    EXECUTE IMMEDIATE
    CONCAT(
       'INSERT INTO ',db,'_profile.',tbl,'_duplicates
        ($Duplicates,',@grplst,')
        SELECT COUNT(*),',@grplst,' FROM ',db,'.',tbl,'
        GROUP BY ',@grplst,'
        HAVING COUNT(*)>1'
       );

       /*export reporting file*/
       SET                                    @file                                          =
CONCAT(REPLACE(dir,'\\','\\\\'),CONCAT(tbl,'_duplicates_',REPLACE(NOW(),':',''),'.csv'));

       EXECUTE IMMEDIATE
    CONCAT('SELECT * INTO OUTFILE \'',@file,'\'
    FIELDS TERMINATED BY \';\'
    OPTIONALLY ENCLOSED BY \'\"\'
    LINES TERMINATED BY \'\\n\'
    FROM ',db,'_profile.',tbl,'_duplicates');
END$$
DELIMITER;
```

7.1.4 Profile

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `profile`(IN `db_profile` VARCHAR(64) CHARSET utf8,
IN `dir_profile` VARCHAR(256) CHARSET utf8)
BEGIN

        SET @cnt_num_profile = 0;
        SET @cnt_tot_profile = 0;

        /* create temporary table*/
        EXECUTE IMMEDIATE CONCAT('DROP TABLE IF EXISTS _tmp');
        EXECUTE IMMEDIATE CONCAT(
    'CREATE TABLE _tmp (
         id BIGINT NOT NULL AUTO_INCREMENT,
                name VARCHAR(64) NULL,
         PRIMARY KEY (Id)
    ) ENGINE = InnoDB');

        /*get tables in database*/
        INSERT  INTO  _tmp  (name)  SELECT  TABLE_NAME  FROM  information_schema.TABLES  WHERE
TABLE_SCHEMA=db_profile;

        /*get total number of tables*/
        SELECT COUNT(*) INTO @cnt_tot_profile FROM _tmp;

    WHILE @cnt_num_profile < @cnt_tot_profile DO
        SET @cnt_num_profile = @cnt_num_profile+1;

                /* get table name*/
                SELECT name INTO @table_name FROM _tmp WHERE id=@cnt_num_profile;

                /* call procedures */
        CALL `duplicates`(db_profile, @table_name, dir_profile);
                CALL `content`(db_profile, @table_name, dir_profile);
         CALL `structure`(db_profile, @table_name, dir_profile);

    END WHILE;

        EXECUTE IMMEDIATE CONCAT('DROP TABLE IF EXISTS _tmp');
END$$
DELIMITER;
```

7.1.5 Distinct occurrences overall

```
DELIMITER $$
CREATE  DEFINER=`root`@`localhost`  PROCEDURE  `distinctinstanceoverall`(IN  `db`  VARCHAR(64)
CHARSET utf8, IN `tbl` VARCHAR(64) CHARSET utf8, IN `col` VARCHAR(64) CHARSET utf8)
EXECUTE IMMEDIATE CONCAT(
       'SELECT
       ',col,',
       COUNT(*) AS Number,
       ROUND((COUNT(*) * 100 / SUM(COUNT(*)) OVER()),1) AS Percent
       FROM ',db,'.',tbl,'
       GROUP BY (CAST(',col,' AS CHAR CHARACTER SET utf8) COLLATE utf8_bin)
    ORDER BY Percent DESC'
    )$$
DELIMITER;
```

7.1.6 Distinct occurrences per group

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `distinctinstancegroup`(IN `db` VARCHAR(64) CHARSET
utf8, IN `tbl` VARCHAR(64) CHARSET utf8, IN `colprobe` VARCHAR(64) CHARSET utf8, IN `colgroup`
VARCHAR(64) CHARSET utf8)
EXECUTE IMMEDIATE
CONCAT(
'SELECT
       ',colgroup,' AS ',colgroup,',
       COUNT(DISTINCT (CAST(',colprobe,' AS CHAR CHARACTER SET utf8) COLLATE utf8_bin)) AS
Instances,
       ROUND((COUNT(DISTINCT (CAST(',colprobe,' AS CHAR CHARACTER SET utf8) COLLATE utf8_bin))
* 100 / SUM(COUNT(DISTINCT (CAST(',colprobe,' AS CHAR CHARACTER SET utf8) COLLATE utf8_bin)))
OVER()),1) AS I_Percent,
    COUNT(*) AS Records,
    ROUND((COUNT(*) * 100 / SUM(COUNT(*)) OVER()),1) AS R_Percent
       FROM ',db,'.',tbl,'
       GROUP BY (CAST(',colgroup,' AS CHAR CHARACTER SET utf8) COLLATE utf8_bin)'
)$$
DELIMITER;
```