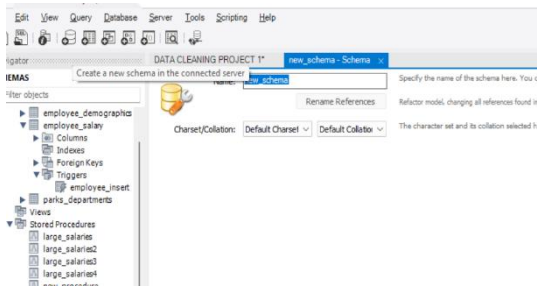
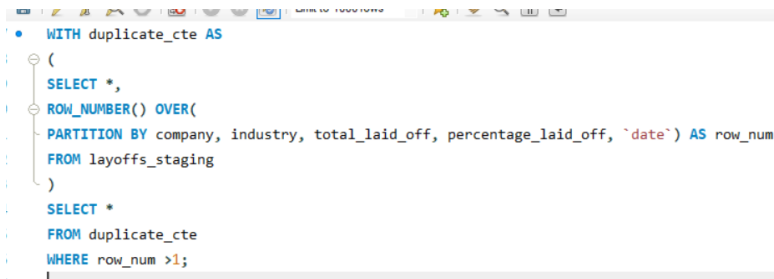


WORLD LAYOFFS 2021-2023

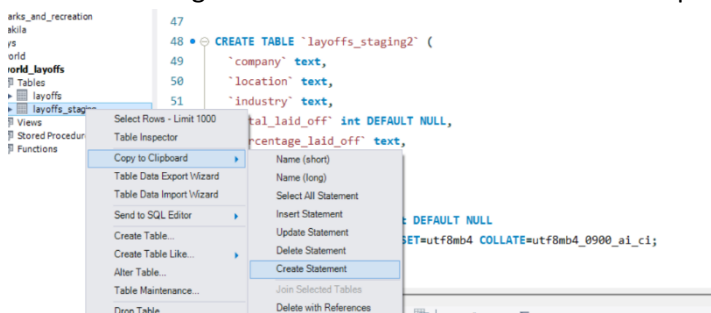
1. Creating new database (new schema) with existing excel (csv) file.



2. Creating new table to be able to clean the data for the future project (Project 2).
3. Check and remove duplicates.
4. Standardize the Data (if there are any issues).
5. Check for NULL or Blank values.
6. Remove Any columns (unnecessary columns).



- 6.1. We create another table (layoffs_staging) as we'll be changing our data a lot and we don't want to lose it.
- 6.2. It is really important (when there are more columns) to check some of data if that is really a duplicate. Example here – **“not including all the columns showed that some rows were taken as duplicates by mistake** (some columns had different data for those rows)”. So it's important to include **all the columns** when checking table for duplicates.
7. Creating additional table to be able to delete duplicates and not lost data (MySQL*).



- 7.1. Then we use **DELETE** statement to delete duplicates from layoffs_staging2 table. We'll delete **row_num** column later. (*SQL Editor settings to turn off Safe mode)
8. Using **TRIM** to delete spaces in **company** column.

```
SELECT company, TRIM(company)
FROM layoffs_staging2;
```

```
UPDATE layoffs_staging2
SET company = TRIM(company);
```

9. Used **UPDATE** to clean data in **Industry** column. Proceeding the checking rest of the columns.

```

SELECT *
FROM layoffs_staging2
WHERE industry LIKE 'Crypto%'
ORDER BY industry;

UPDATE layoffs_staging2
SET industry = 'Crypto'
WHERE industry LIKE 'Crypto%';

```

10. Cleaning **countries** in two ways (we had “United States” and “United States.” (with a dot)).

```

SELECT DISTINCT country
FROM layoffs_staging2
WHERE country LIKE 'United St%';

UPDATE layoffs_staging2
SET country = 'United States'
WHERE country = 'United States.';

```

```

SELECT DISTINCT country, TRIM(TRAILING '.' FROM country)
FROM layoffs_staging2
ORDER BY 1;

```

11. Now we change data type of column **date** from Text to String, but it'll still keep the data type as Text.

11.1. To change it we use **ALTER TABLE**.

The screenshot shows a SQL IDE interface with a schema tree on the left and a query editor on the right. The schema tree shows a table named 'layoffs_staging2' with columns: stage, country, funds_raised_mil, row_num, date, and industry. The 'date' column is highlighted, and its definition is shown as 'date text'. The query editor shows the following SQL code:

```

103 • SELECT `date`
104 • FROM layoffs_staging2;

105
106 • UPDATE layoffs_staging2
107 • SET `date` = STR_TO_DATE(`date`, '%m/%d/%Y');

109 • ALTER TABLE layoffs_staging2
110 • MODIFY COLUMN `date` DATE;

```

The 'Result Grid' shows the output of the first query, displaying a list of dates. The 'Action Output' shows the execution of the ALTER TABLE statement, indicating that the column 'date' has been successfully modified to 'DATE'.

12. Some of **Industries** are Blank or NULL. If there are other data for those, we might need to populate it (check if it's useless for us or no beforehand).

The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor shows the following SQL code:

```

123
124 • SELECT *
125 • FROM layoffs_staging2
126 • WHERE industry IS NULL
127 • OR industry = '';

```

The 'Result Grid' shows the output of the query, displaying a list of rows where the 'industry' column is NULL or blank. The columns shown are: company, location, industry, total_laid_off, percentage_laid_off, and date.

12.1. Before we populate (change) anything that has NULLs or BLANKs we need to check (better on JOINS, or SELF JOINS in this case) if it's true that there are NULLs and BLANKs for everything or we have information perhaps in other rows. Then we make an **UPDATE**.

```

131 • SELECT t1.company, t1.location, t1.industry, t2.industry
132 FROM layoffs_staging2 t1
133 JOIN layoffs_staging2 t2
134     ON t1.company = t2.company
135     AND t1.location = t2.location
136 WHERE (t1.industry IS NULL OR t1.industry = '')
137 AND t2.industry IS NOT NULL;
138
139

```

```

128
129 • UPDATE layoffs_staging2
130 SET industry = NULL
131 WHERE industry = '';
132
133 • SELECT t1.company, t1.location, t1.industry, t2.industry
134 FROM layoffs_staging2 t1
135 JOIN layoffs_staging2 t2
136     ON t1.company = t2.company
137     AND t1.location = t2.location
138 WHERE (t1.industry IS NULL OR t1.industry = '')
139 AND t2.industry IS NOT NULL;
140
141 • UPDATE layoffs_staging2 t1
142 JOIN layoffs_staging2 t2
143     ON t1.company = t2.company
144 SET t1.industry = t2.industry
145 WHERE t1.industry IS NULL
146 AND t2.industry IS NOT NULL;
147

```

12.2. Here we also used JOIN to find what data was empty (blank) and what was having NULL so to be able to populate **the blanks with nulls at first**, and then **make an update with the existing data**.

13. Now we found that there are a lot of rows that **have no lay offs and no percentage of lay offs**, which means such rows are not really useful for our project as we can't populate, find this information, it was just not added there initially. So we can **DELETE** it.

```

• SELECT *
FROM layoffs_staging2
WHERE total_laid_off IS NULL
AND percentage_laid_off IS NULL;

• DELETE
FROM layoffs_staging2
WHERE total_laid_off IS NULL
AND percentage_laid_off IS NULL;

```

company	location	industry	total_laid_off	percentage_laid_off
nc.	Toronto	Transportation	NULL	NULL
) Thieves	Los Angeles	Retail	NULL	NULL
olade	Seattle	Healthcare	NULL	NULL
a	Toronto	Support	NULL	NULL
ara	SF Bay Area	Travel	NULL	NULL

13.1. We also have column we added earlier to look for duplicates that we don't need anymore and we can **DROP** it.

```

ALTER TABLE layoffs_staging2
DROP COLUMN row_num;

```

14. Additionally, I wanted to clean the **NULLs** data rows from our database (in columns **total**, **percentage**, **stage** and **funds**), however without losing all possible aggregation functions we can use in the future and without changing column's data types.

```

177 • SELECT
178     company, location, industry,
179     CONCAT(IFNULL(total_laid_off, 'No data')) AS total_laid_off_display,
180     CONCAT(IFNULL(percentage_laid_off, 'No data')) AS percentage_laid_off_display, -- as we don't want to change the type of column (from INT to STR)
181     `date`, stage, country, funds_raised_millions
182 FROM layoffs_staging2;
183

```

15. For the future, just wanted to learn for myself, how can I quickly get all the columns I have in the table (as there can be way more than 3 for instance) and how to create a default view.

```

• SELECT GROUP_CONCAT(column_name ORDER BY ordinal_position) -- how to choose all your column names and create a new view if needed
FROM information_schema.columns
WHERE table_schema = 'world_layoffs'
AND table_name = 'layoffs_staging2';

• CREATE VIEW layoffs_staging2_revised AS
SELECT company, location, industry, percentage_laid_off, `date`, stage, country, funds_raised_millions
FROM layoffs_staging2;

• SELECT *
FROM layoffs_staging2_revised;

```

16. It, however, makes no harm to change from whatever type was in column **percentage** to **FLOAT**.

```

ALTER TABLE layoffs_staging2
MODIFY COLUMN percentage_laid_off FLOAT;

```

17. Next, I wanted to display how the **NULLS** disappeared from the report (replaced by **No data**) and kept all rows in table...

17.1. ...using previous **display** functionality with **CONCAT** & **IFNULL** and now **UNION ALL** to show the short report.

```

-- 1. Summary:
203
204 • SELECT
205     "DATA CLEANING SUMMARY" AS company,
206     "NULLS replaced with 'No data'" AS location,
207     "AS industry",
208     CONCAT(
209         (SELECT COUNT(*) FROM layoffs WHERE total_laid_off IS NULL),
210         " → 0"
211     ) AS total_laid_off_display,
212     CONCAT(
213         (SELECT COUNT(*) FROM layoffs WHERE percentage_laid_off IS NULL),
214         " → 0"
215     ) AS percentage_laid_off_display,
216     "AS 'date'",
217     CONCAT(
218         (SELECT COUNT(*) FROM layoffs WHERE `date` IS NULL),
219         " → 0"
220     ) AS `date`_display,
221     "AS stage",
222     CONCAT(
223         (SELECT COUNT(*) FROM layoffs WHERE stage IS NULL),
224         " → 0"
225     ) AS stage_display,
226     "AS country",
227     CONCAT(
228         (SELECT COUNT(*) FROM layoffs WHERE country IS NULL),
229         " → 0"
230     ) AS country_display,
231     "AS funds_raised_millions"
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

company	location	industry	total_laid_off_display	percentage_laid_off_display	date	stage	country	funds_raised_millions
"DATA CLEANING SUMMARY"	"NULLS replaced with 'No data'"		740 → 0	785 → 0		6 → 0		209 → 0
Company	Location	Industry	Total Laid Off	Percentage Laid Off	Date	Stage	Country	Funds Raised
Included Health	SF Bay Area	Healthcare	No data	0.06	2022-07-25	Series E	United States	272
\$Open	Dublin	Marketing	9	0.09	2022-11-17	Series A	Ireland	35
#Paid	Toronto	Marketing	19	0.17	2023-01-27	Series B	Canada	21

```

-- 2. Header row
SELECT
    'Company' AS company,
    'Location' AS location,
    'Industry' AS industry,
    'Total Laid Off' AS total_laid_off_display,
    'Percentage Laid Off' AS percentage_laid_off_display,
    'Date' AS `date`,
    'Stage' AS stage,
    'Country' AS country,
    'Funds Raised ($M)' AS funds_raised_millions
UNION ALL
-- 3. Your cleaned data – all rows kept, NULLs shown as 'No data'
SELECT
    company,
    location,
    industry,
    IFNULL(total_laid_off, 'No data') AS total_laid_off_display,
    IFNULL(percentage_laid_off, 'No data') AS percentage_laid_off_display,
    `date`,
    stage,
    country,
    funds_raised_millions
FROM layoffs_staging2;
UNION ALL

```

18. Finally, I thought, why not to add some **data quality** measurement. And so...

```
258 • ALTER TABLE layoffs_staging2
259   ADD COLUMN data_quality_score INT DEFAULT 0;
260
261 • UPDATE layoffs_staging2
262   SET data_quality_score =
263     (CASE WHEN total_laid_off IS NOT NULL THEN 1 ELSE 0 END) +
264     (CASE WHEN percentage_laid_off IS NOT NULL THEN 1 ELSE 0 END) +
265     (CASE WHEN industry IS NOT NULL AND industry != '' THEN 1 ELSE 0 END) +
266     (CASE WHEN stage IS NOT NULL AND stage != 'Unknown' THEN 1 ELSE 0 END) +
267     (CASE WHEN funds_raised_millions IS NOT NULL THEN 1 ELSE 0 END);
268
269 • SELECT data_quality_score, COUNT(*)
270   FROM layoffs_staging2
271  GROUP BY 1
272  ORDER BY 1 DESC;
273
274 • SELECT MAX(data_quality_score), MIN(data_quality_score)
275   FROM layoffs_staging2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: |

	data_quality_score	COUNT(*)
▶	5	954
	4	829
	3	178
	2	34

19. Ahhh, and **created a view** just to simply underscore few last steps in returning one short command then

```
281 • CREATE VIEW vw_layoffs_clean AS
282   SELECT
283     company,
284     TRIM(company) AS company_clean,
285     location,
286     industry,
287     IFNULL(total_laid_off, 'No data') AS total_laid_off_display,
288     IFNULL(percentage_laid_off, 'No data') AS percentage_laid_off_display,
289     `date`,
290     YEAR(`date`) AS year,
291     MONTH(`date`) AS month,
292     IFNULL(stage, 'Unknown') AS stage,
293     country,
294     IFNULL(funds_raised_millions, 'No data') AS funds_raised_millions,
295     data_quality_score
296   FROM layoffs_staging2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: | Fetch rows: |

	company	company_clean	location	industry	total_laid_off_display	percentage_laid_off_display	date	year	month	stage	country	funds_raised_millions	data_quality_score
▶	Included Health	Included Health	SF Bay Area	Healthcare	No data	0.06	2022-07-25	2022	7	Series E	United States	272	4
	&Open	&Open	Dublin	Marketing	9	0.09	2022-11-17	2022	11	Series A	Ireland	35	5
	#Paid	#Paid	Toronto	Marketing	19	0.17	2023-01-27	2023	1	Series B	Canada	21	5
	100 Thieves	100 Thieves	Los Angeles	Consumer	12	No data	2022-07-13	2022	7	Series C	United States	120	4
	10X Genomics	10X Genomics	SF Bay Area	Healthcare	100	0.08	2022-08-04	2022	8	Post-IPO	United States	242	5

20. Some calculation of difference cleaned data...

```
-- some calculation of difference removed dups
SELECT
  (SELECT COUNT(*) FROM layoffs) AS initial_data,
  (SELECT COUNT(*) FROM layoffs_staging2) AS after_cleaning_data,
  (SELECT COUNT(*) FROM layoffs) - (SELECT COUNT(*) FROM layoffs_staging2) AS removed_dups;
```

21. And, which might become handy in the future, some **automatization** to check for.

```
316 DELIMITER //
317 • CREATE PROCEDURE CleanLayoffsData()
318 BEGIN
319   DELETE t1 FROM layoffs_staging2 t1
320   INNER JOIN layoffs_staging2 t2
321   WHERE
322     t1.company = t2.company AND
323     t1.location = t2.location AND
324     t1.`date` = t2.`date` AND
325     t1.total_laid_off = t2.total_laid_off;
326
327   UPDATE layoffs_staging2
328   SET country = 'United States'
329   WHERE country LIKE 'United States%';
330
331   UPDATE layoffs_staging2 t1
332   JOIN layoffs_staging2 t2 ON t1.company = t2.company
333   SET t1.industry = t2.industry
334   WHERE t1.industry IS NULL AND t2.industry IS NOT NULL;
335 END //
336 DELIMITER ;
337
```

- Did the whole data cleaning process: removed duplicates, standardized text, cleared NULLs and converted data types.
- "Added my own measurement system to verify data quality and assign the score (data_quality_score 0-5).
- Created analytical view and report that summarizes data cleaning process and the whole project.