

SQL Project: World Layoffs

Data Cleaning

The project leverages MySQL to dive into the data on worldwide employee layoffs across different industries.

The project is aimed at:

Writing effective SQL queries to clean, manipulate and transform the data to derive meaningful insights.


```

14  -- Creating a copy of the raw data
15  • create table layoffs_staging like layoffs;
16
17  • insert layoffs_staging
18    select * from layoffs;
19
20  • select * from layoffs_staging;

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows:

	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
▶	Atlassian	Sydney	Other	500	0.05	3/6/2023	Post-IPO	Australia	210
	SiriusXM	New York City	Media	475	0.08	3/6/2023	Post-IPO	United States	525
	Alerzo	Ibadan	Retail	400	NULL	3/6/2023	Series B	Nigeria	16
	UpGrad	Mumbai	Education	120	NULL	3/6/2023	Unknown	India	631
	Loft	Sao Paulo	Real Estate	340	0.15	3/3/2023	Unknown	Brazil	788
	Embark Trucks	SF Bay Area	Transportation	230	0.7	3/3/2023	Post-IPO	United States	317
	Lendi	Sydney	Real Estate	100	NULL	3/3/2023	Unknown	Australia	59
	UserTesting	SF Bay Area	Marketing	63	NULL	3/3/2023	Acquired	United States	152
	Airbnb	SF Bay Area		30	NULL	3/3/2023	Post-IPO	United States	6400

layoffs_staging 5 ×

Output



Action Output

#	Time	Action	Message
✓ 1	15:35:23	select * from layoffs_staging LIMIT 0, 1000	1000 row(s) returned

Removing Duplicate Records

```
27 -- 1. Checking for duplicate records using ROW_NUMBER & OVER functions as there is no unique key in the dataset.
28 • with duplicate_cte as
29 (
30   select *,
31   ROW_NUMBER() over(
32     PARTITION BY company, location, industry,
33     total_laid_off, percentage_laid_off, 'date', stage, country, funds_raised_millions) AS row_num
34   from layoffs_staging
35 )
36 select *
37 from duplicate_cte
38 where row_num >1;
39
```

	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions	row_num
▶	Better.com	New York City	Real Estate	NULL	NULL	8/26/2022	Unknown	United States	905	2
	Casper	New York City	Retail	NULL	NULL	9/14/2021	Post-IPO	United States	339	2
	Cazoo	London	Transportation	750	0.15	6/7/2022	Post-IPO	United Kingdom	2000	2
	Elemy	SF Bay Area	Healthcare	NULL	NULL	12/5/2022	Series B	United States	323	2
	ExtraHop	Seattle	Security	NULL	NULL	4/22/2020	Series C	United States	61	2
	Hibob	Tel Aviv	HR	70	0.3	3/30/2020	Series A	Israel	45	2

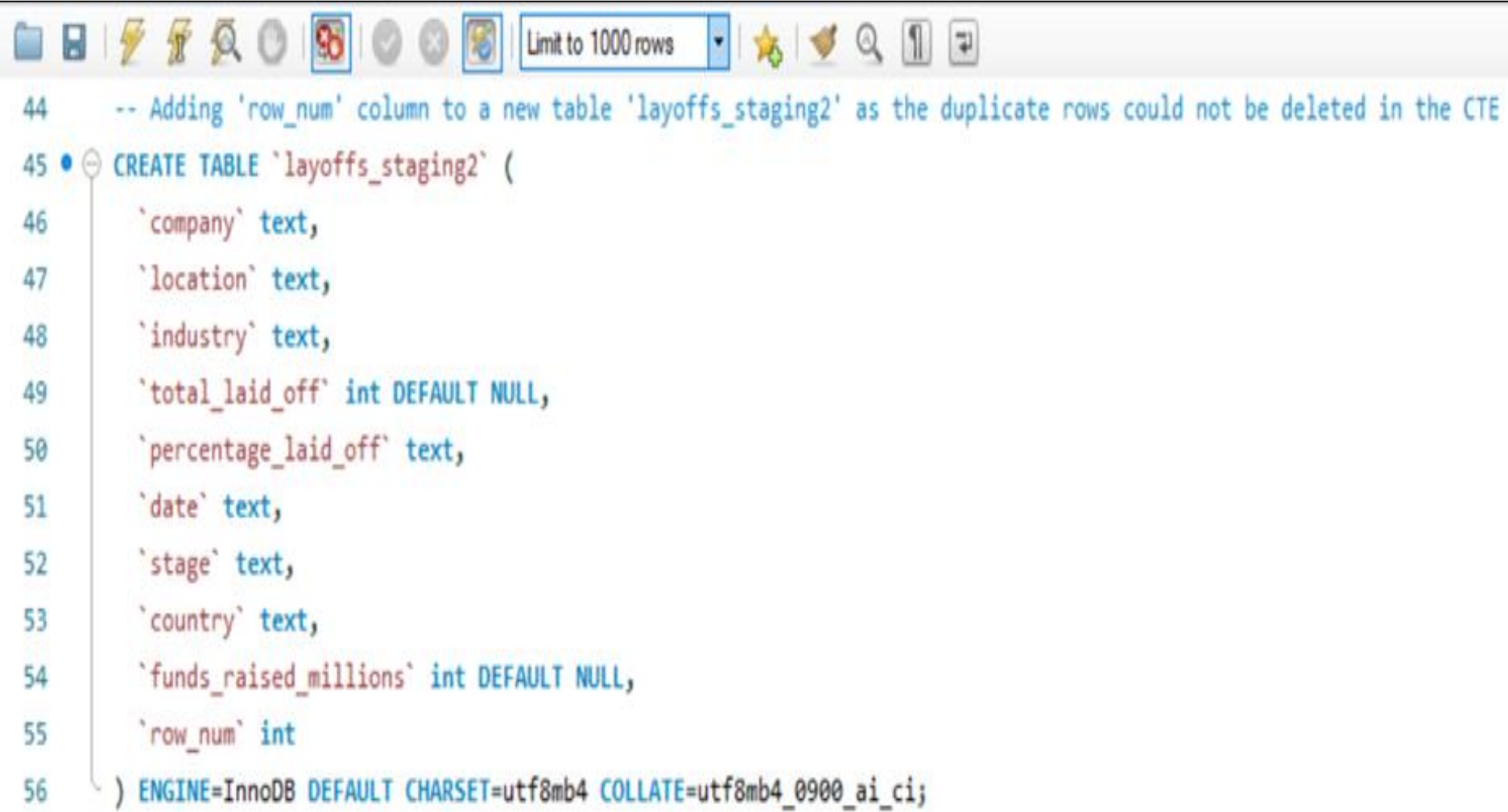
Result 7 x

Output



Action Output

#	Time	Action	Message
✓ 1	15:42:48	with duplicate_cte as (select *, ROW_NUMBER() over(PARTITION BY company, lo...	22 row(s) returned



The screenshot shows a SQL IDE window with a toolbar at the top. The toolbar includes icons for file operations (folder, save, undo, redo, find, replace), a 'Limit to 1000 rows' dropdown, and other utility icons. The main text area contains a SQL script with line numbers 44 through 56. Line 44 is a comment explaining the purpose of the new table. Line 45 starts the CREATE TABLE statement for 'layoffs_staging2'. Lines 46 through 55 list the columns: 'company' (text), 'location' (text), 'industry' (text), 'total_laid_off' (int, DEFAULT NULL), 'percentage_laid_off' (text), 'date' (text), 'stage' (text), 'country' (text), 'funds_raised_millions' (int, DEFAULT NULL), and 'row_num' (int). Line 56 completes the statement with the engine and charset/colate settings.

```
44 -- Adding 'row_num' column to a new table 'layoffs_staging2' as the duplicate rows could not be deleted in the CTE
45 • CREATE TABLE `layoffs_staging2` (
46     `company` text,
47     `location` text,
48     `industry` text,
49     `total_laid_off` int DEFAULT NULL,
50     `percentage_laid_off` text,
51     `date` text,
52     `stage` text,
53     `country` text,
54     `funds_raised_millions` int DEFAULT NULL,
55     `row_num` int
56 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Limit to 1000 rows

```

57  -- Inserting data into 'layoffs_staging2' table and adding 'row_num' column to DELETE duplicate records.
58  • select * from layoffs_staging2
59      WHERE row_num >1;
60
61  • insert into layoffs_staging2
62      select *,
63      ROW_NUMBER() over(
64      PARTITION BY company, location, industry,
65      total_laid_off, percentage_laid_off, 'date', stage, country, funds_raised_millions) AS row_num
66      from layoffs_staging;
67
68  • delete from layoffs_staging2
69      WHERE row_num >1;
70
71  • select * from layoffs_staging2;

```

Result Grid

Filter Rows:

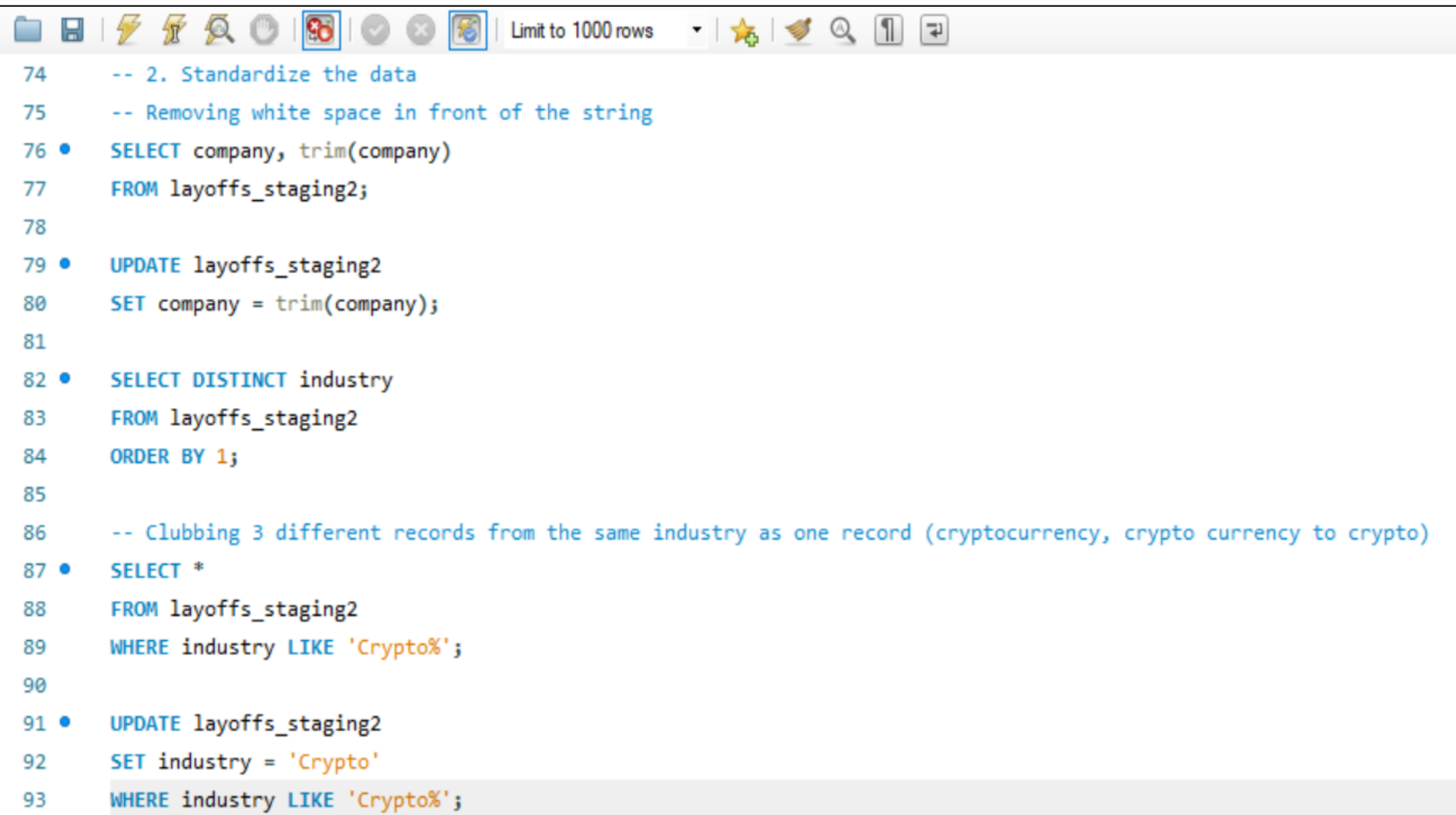
Export:

Wrap Cell Content:


Fetch rows:

	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
▶	Included Health	SF Bay Area	Healthcare	NULL	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
	100 Thieves	Los Angeles	Consumer	12	NULL	2022-07-13	Series C	United States	120
	10X Genomics	SF Bay Area	Healthcare	100	0.08	2022-08-04	Post-IPO	United States	242

Standardize the Data

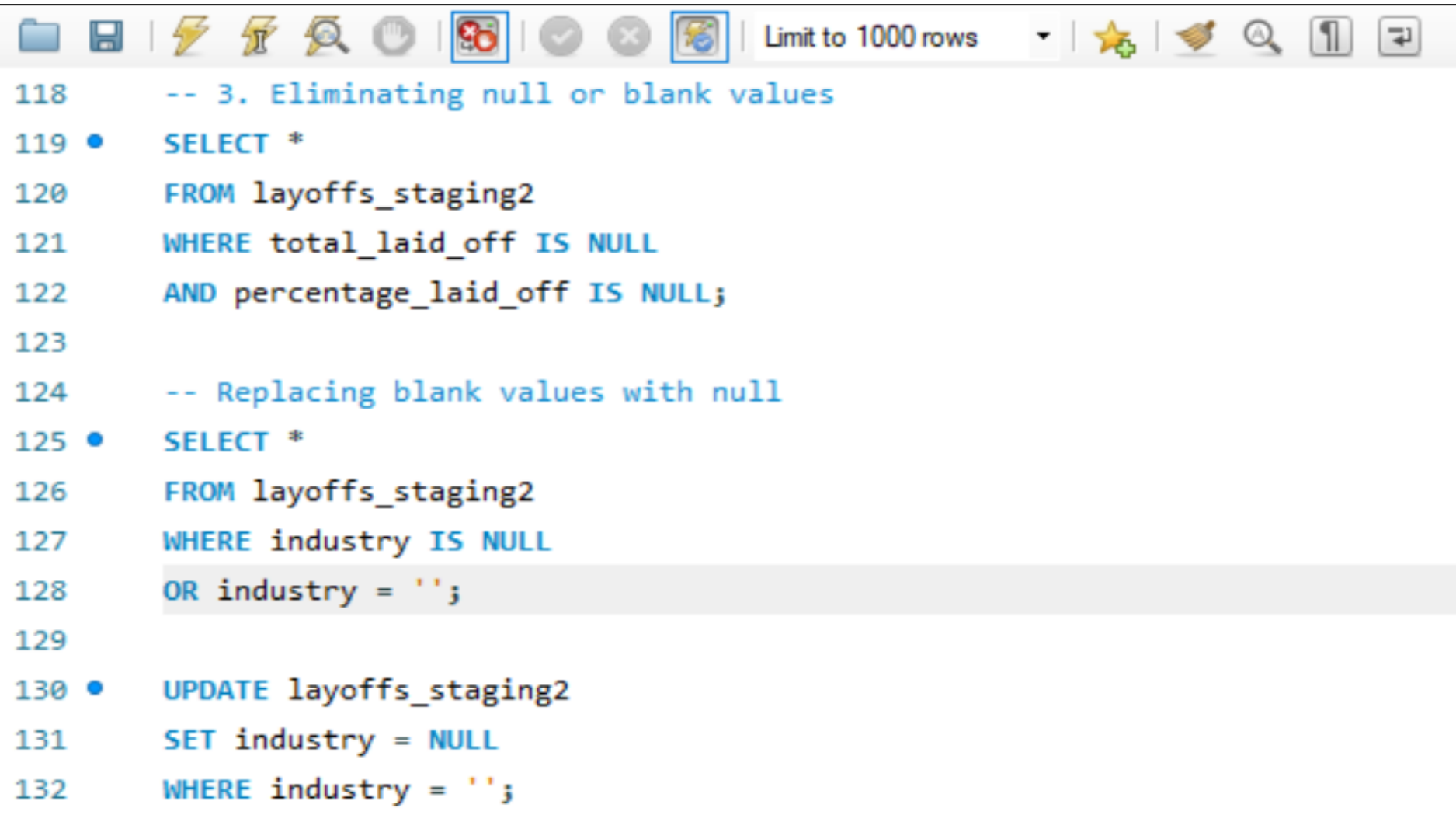


```
74  -- 2. Standardize the data
75  -- Removing white space in front of the string
76  • SELECT company, trim(company)
77  FROM layoffs_staging2;
78
79  • UPDATE layoffs_staging2
80  SET company = trim(company);
81
82  • SELECT DISTINCT industry
83  FROM layoffs_staging2
84  ORDER BY 1;
85
86  -- Clubbing 3 different records from the same industry as one record (cryptocurrency, crypto currency to crypto)
87  • SELECT *
88  FROM layoffs_staging2
89  WHERE industry LIKE 'Crypto%';
90
91  • UPDATE layoffs_staging2
92  SET industry = 'Crypto'
93  WHERE industry LIKE 'Crypto%';
```


```
105 -- Removing trailing '.' from a duplicate record (United States, United States.) to a single unique record (United States)
106 • SELECT DISTINCT country, trim(trailing '.' from country)
107 FROM layoffs_staging2
108 ORDER BY 1;
109
110 • UPDATE layoffs_staging2
111 SET country = trim(trailing '.' from country)
112 WHERE country LIKE 'United States%';
113
114 -- Converting text to date format
115 • SELECT `date`,
116 str_to_date(`date`, '%m/%d/%Y')
117 FROM layoffs_staging2;
118
119 • UPDATE layoffs_staging2
120 SET `date` = str_to_date(`date`, '%m/%d/%Y');
121
122 • SELECT `date` FROM layoffs_staging2;
123
124 -- Changing data type to date
125 • ALTER TABLE layoffs_staging2
126 MODIFY COLUMN `date` DATE;
```


Eliminating Null or Blank Values



The screenshot shows a SQL IDE window with a toolbar at the top. The toolbar includes icons for file operations (folder, save), execution (lightning bolt, play), search (magnifying glass), and other utilities. A dropdown menu is set to "Limit to 1000 rows". The main area displays SQL code with line numbers 118 through 132. The code consists of three SQL statements: a SELECT query to find rows with null values in 'total_laid_off' and 'percentage_laid_off', a comment about replacing blank values with null, and an UPDATE query to set 'industry' to null for rows where 'industry' is blank.

```
118      -- 3. Eliminating null or blank values
119  ●   SELECT *
120      FROM layoffs_staging2
121      WHERE total_laid_off IS NULL
122      AND percentage_laid_off IS NULL;
123
124      -- Replacing blank values with null
125  ●   SELECT *
126      FROM layoffs_staging2
127      WHERE industry IS NULL
128      OR industry = '';
129
130  ●   UPDATE layoffs_staging2
131      SET industry = NULL
132      WHERE industry = '';
```



```
129      -- Cross-checking company within the same industry with different records
130      -- (2 records of Airbnb in Travel industry and other record as null)
131 •    SELECT *
132      FROM layoffs_staging2
133      WHERE company = 'Airbnb';
134
135      -- Performing a self-join to fetch the 'industry' column which has null values for a company with the same name
136 •    SELECT t1.industry, t2.industry
137      FROM layoffs_staging2 t1
138      JOIN layoffs_staging2 t2
139          ON t1.company = t2.company
140      WHERE (t1.industry IS NULL OR t1.industry = '')
141      AND t2.industry IS NOT NULL;
142
143      -- Replacing the null values
144 •    UPDATE layoffs_staging2 t1
145      JOIN layoffs_staging2 t2
146          ON t1.company = t2.company
147      SET t1.industry = t2.industry
148      WHERE t1.industry IS NULL
149      AND t2.industry IS NOT NULL;
```

Limit to 1000 rows

```

151      -- Checking for null values in 'total_laid_off' & 'percentage_laid_off' columns and deleting the same
152 •    SELECT *
153      FROM layoffs_staging2
154      WHERE total_laid_off IS NULL
155      AND percentage_laid_off IS NULL;
156
157 •    DELETE
158      FROM layoffs_staging2
159      WHERE total_laid_off IS NULL
160      AND percentage_laid_off IS NULL;
161
162      -- 4. Removing column that is not required
163 •    ALTER TABLE layoffs_staging2
164      DROP COLUMN row_num;
165
166      -- Final clean data
167 •    SELECT * FROM layoffs_staging2;
168

```

Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Fetch rows:

	company	location	industry	total_laid_off	percentage_laid_off	date	stage	country	funds_raised_millions
▶	Included Health	SF Bay Area	Healthcare	NULL	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
	100 Thieves	Los Angeles	Consumer	12	NULL	2022-07-13	Series C	United States	120
	10X Genomics	SF Bay Area	Healthcare	100	0.08	2022-08-04	Post-IPO	United States	242