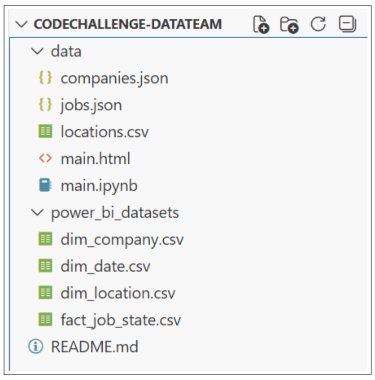
Fionn O’Reilly – GOHiring Data Team Code Challenge

How Data Was Loaded

The two json files and csv file were loaded using pandas in a Jupyter Notebook in Visual Studio Code. The ipynb file was placed in the same folder as the json and csv files.



```
CODECHALLENGE-DATATEAM
  data
    companies.json
    jobs.json
    locations.csv
    main.html
    main.ipynb
  power_bi_datasets
    dim_company.csv
    dim_date.csv
    dim_location.csv
    fact_job_state.csv
  README.md
```

main.ipynb contains the main code and logic with main.html being an export of that file.

The modelled and transformed datasets were written to csv files in the power_bi_datasets folder. These files were imported into Power BI for visualisations.

Data Quality Issues Found

Establishment Date in the companies data needed to be converted from unix timestamp to date.

```python
companies = pd.read_json('companies.json')

# Convert Establishment Date from unix timestamp to date
companies["Establishment Date"] = pd.to_datetime(companies['Establishment Date'], unit='ms')
```

The state field in the jobs data had typos that were corrected with a case statement.

```sql
,CASE
    WHEN j.state IN ('posted','expired','cancelled') THEN j.state
    WHEN j.state = 'osted' THEN 'posted'
    WHEN j.state = 'canceled' THEN 'cancelled'
    ELSE j.state
END AS state
```

Column headers in the companies table were renamed to remove white space and match the naming convention of the other fields.

Additional Notes

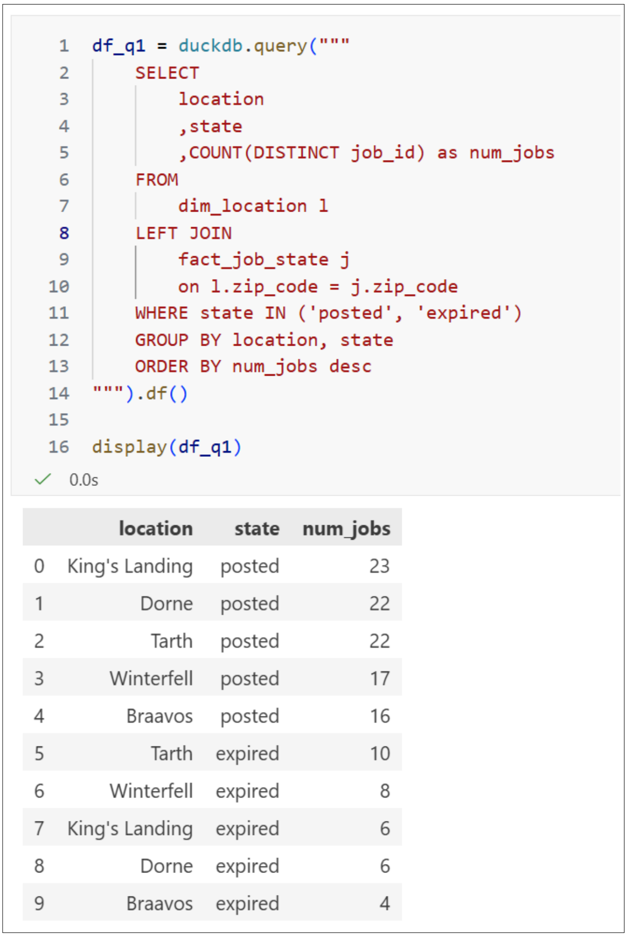
The data was transformed and modelled into fact and dimension tables which were then used to answer each question. An entity relationship diagram can be seen further down the document. The four tables are;

- fact_job_state
- dim_company
- dim_location
- dim_date

1. Which location currently has the most jobs in either the active or expired state?

The query and output show that the location with the most active jobs is King's Landing with 23 jobs posted, and the location with the most expired jobs is Tarth, with 10 jobs expired.

To return only the top locations for each state, a window function could be used to rank each location by count of job_id, partitioning by state.



```
df_q1 = duckdb.query("""
    SELECT
        location
        ,state
        ,COUNT(DISTINCT job_id) as num_jobs
    FROM
        dim_location l
    LEFT JOIN
        fact_job_state j
        on l.zip_code = j.zip_code
    WHERE state IN ('posted', 'expired')
    GROUP BY location, state
    ORDER BY num_jobs desc
""").df()

display(df_q1)
```

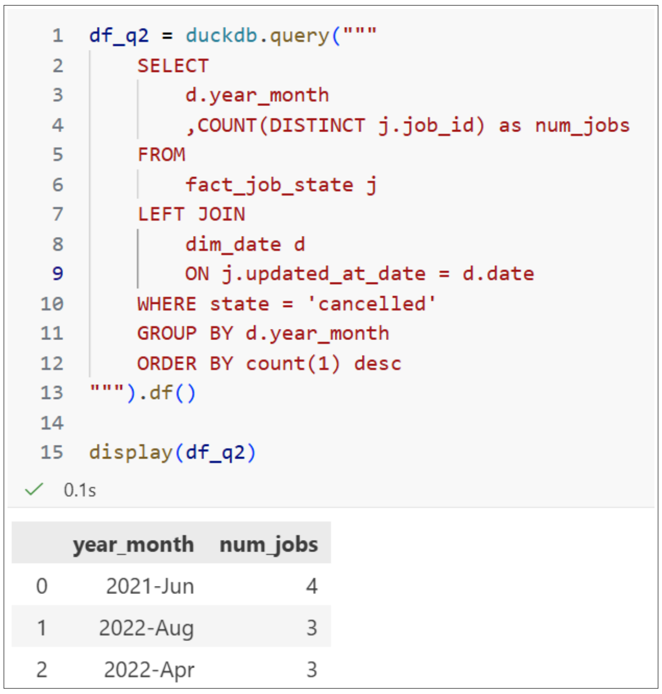
✓ 0.0s

	location	state	num_jobs
0	King's Landing	posted	23
1	Dorne	posted	22
2	Tarth	posted	22
3	Winterfell	posted	17
4	Braavos	posted	16
5	Tarth	expired	10
6	Winterfell	expired	8
7	King's Landing	expired	6
8	Dorne	expired	6
9	Braavos	expired	4

2. In which month were the most jobs cancelled

The month with the most cancelled jobs is June 2021. The 'year_month' field in the dim_date table was used to aggregate cancelled jobs by month.

The year_month field extracts and concatenates the year and month from the updated_at field.



```
df_q2 = duckdb.query("""
    SELECT
        d.year_month
        ,COUNT(DISTINCT j.job_id) as num_jobs
    FROM
        fact_job_state j
    LEFT JOIN
        dim_date d
        ON j.updated_at_date = d.date
    WHERE state = 'cancelled'
    GROUP BY d.year_month
    ORDER BY count(1) desc
""").df()

display(df_q2)
```

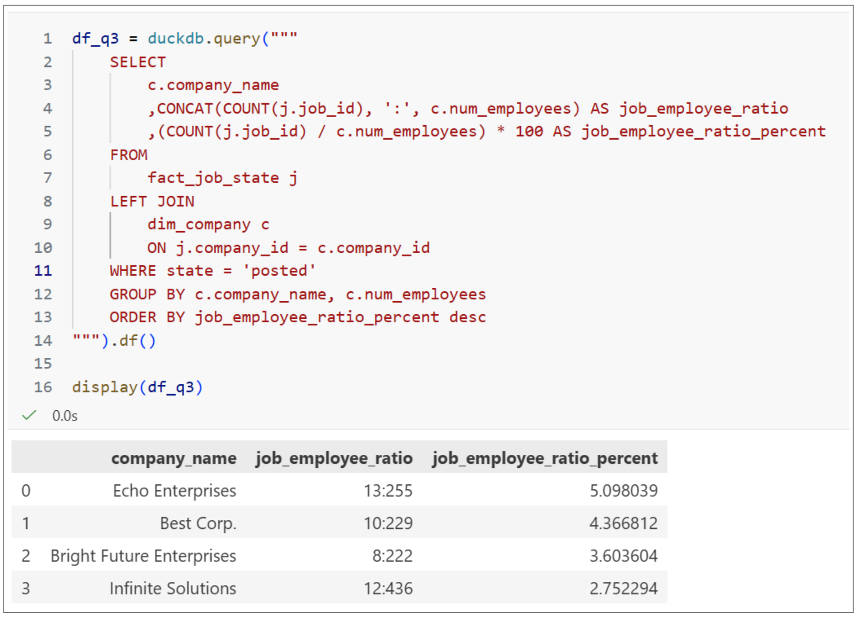
✓ 0.1s

	year_month	num_jobs
0	2021-Jun	4
1	2022-Aug	3
2	2022-Apr	3

3. Which company has the highest ratio of posted jobs to employee count?

By joining the job and company tables and calculating the ratio, the company with the highest posted jobs to employee count ratio is Echo Enterprises with 13:255, or 5.1%.

Improvements to this query could be made by handling the possibility of division by zero, and handling null values. Both could be handled with case statements, or NULLIF/COALESCE functions.



```
df_q3 = duckdb.query("""
    SELECT
        c.company_name
        ,CONCAT(COUNT(j.job_id), ':', c.num_employees) AS job_employee_ratio
        ,(COUNT(j.job_id) / c.num_employees) * 100 AS job_employee_ratio_percent
    FROM
        fact_job_state j
    LEFT JOIN
        dim_company c
        ON j.company_id = c.company_id
    WHERE state = 'posted'
    GROUP BY c.company_name, c.num_employees
    ORDER BY job_employee_ratio_percent desc
""").df()

display(df_q3)
```

✓ 0.0s

	company_name	job_employee_ratio	job_employee_ratio_percent
0	Echo Enterprises	13:255	5.098039
1	Best Corp.	10:229	4.366812
2	Bright Future Enterprises	8:222	3.603604
3	Infinite Solutions	12:436	2.752294

4. Develop the SQL to define a dimensional model schema for this data. Document any design decisions that you make.

The data was modelled into one fact table and three dimension tables.

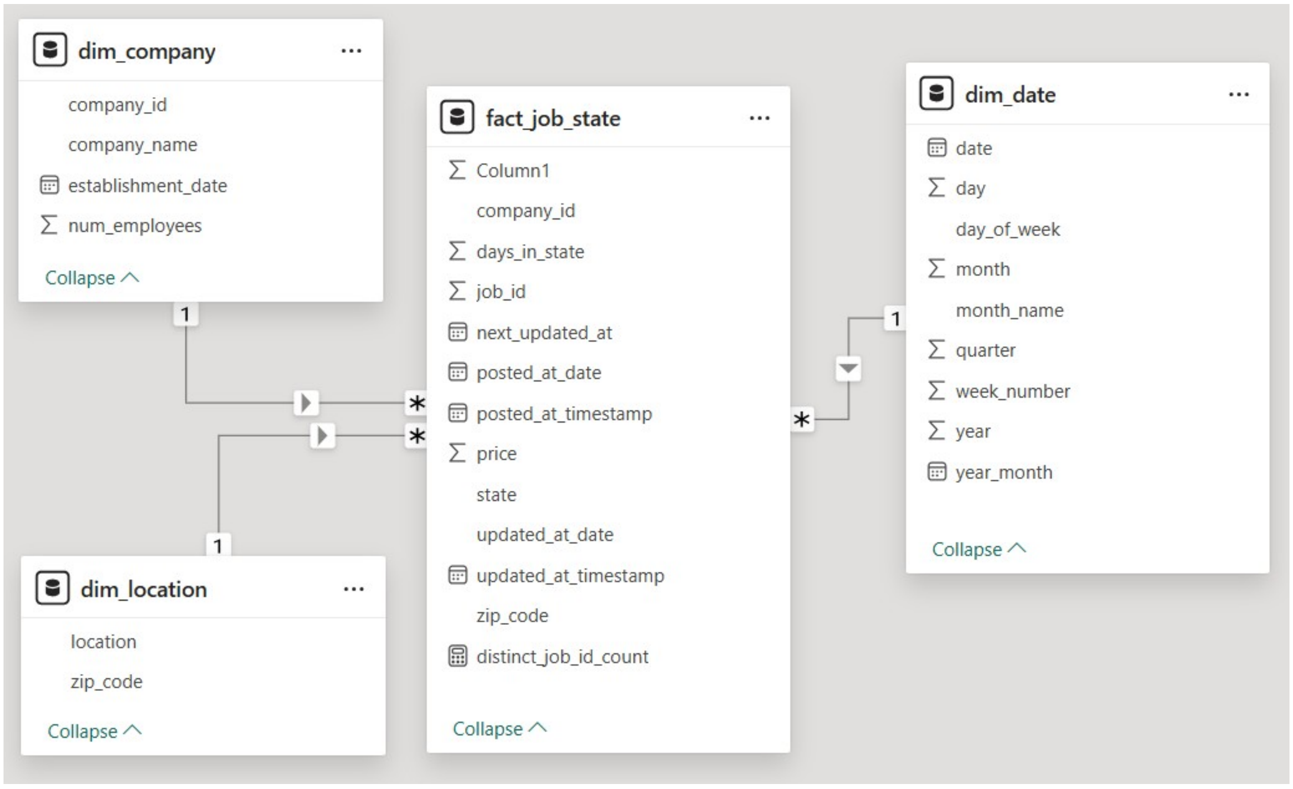
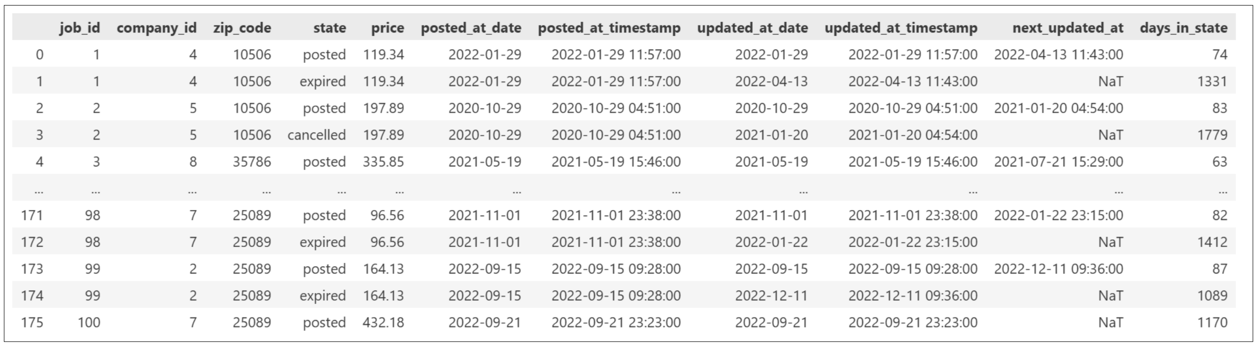


Table 1: fact_job_state

- Contains data about job postings, including state, posted_at, updated_at, price, and zip_code.
- Additional columns were added for updated_at_date and posted_at_date (date types instead of timestamps)
- Data quality issues were fixed
- A ‘days_in_state’ field was added showing the number of days the job has been in each state
- Uses job_id, zip_code, and updated_at_date to join to dimension tables.
- ‘next_updated_at’ was used to calculate days_in_state but would not normally be included in the final table

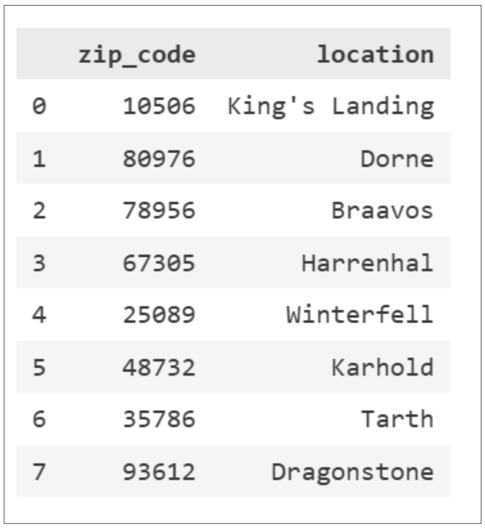
Sample data from fact_job_state table;



	job_id	company_id	zip_code	state	price	posted_at_date	posted_at_timestamp	updated_at_date	updated_at_timestamp	next_updated_at	days_in_state
0	1	4	10506	posted	119.34	2022-01-29	2022-01-29 11:57:00	2022-01-29	2022-01-29 11:57:00	2022-04-13 11:43:00	74
1	1	4	10506	expired	119.34	2022-01-29	2022-01-29 11:57:00	2022-04-13	2022-04-13 11:43:00	NaT	1331
2	2	5	10506	posted	197.89	2020-10-29	2020-10-29 04:51:00	2020-10-29	2020-10-29 04:51:00	2021-01-20 04:54:00	83
3	2	5	10506	cancelled	197.89	2020-10-29	2020-10-29 04:51:00	2021-01-20	2021-01-20 04:54:00	NaT	1779
4	3	8	35786	posted	335.85	2021-05-19	2021-05-19 15:46:00	2021-05-19	2021-05-19 15:46:00	2021-07-21 15:29:00	63
...	...	...	...	...	...	...	...	...	...	...	...
171	98	7	25089	posted	96.56	2021-11-01	2021-11-01 23:38:00	2021-11-01	2021-11-01 23:38:00	2022-01-22 23:15:00	82
172	98	7	25089	expired	96.56	2021-11-01	2021-11-01 23:38:00	2022-01-22	2022-01-22 23:15:00	NaT	1412
173	99	2	25089	posted	164.13	2022-09-15	2022-09-15 09:28:00	2022-09-15	2022-09-15 09:28:00	2022-12-11 09:36:00	87
174	99	2	25089	expired	164.13	2022-09-15	2022-09-15 09:28:00	2022-12-11	2022-12-11 09:36:00	NaT	1089
175	100	7	25089	posted	432.18	2022-09-21	2022-09-21 23:23:00	2022-09-21	2022-09-21 23:23:00	NaT	1170

Table 2: dim_location

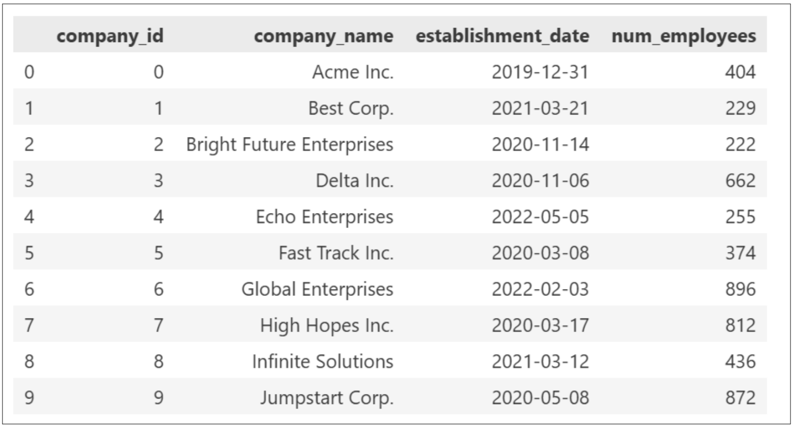
- Kept as is with zip_code and location fields
- Location tables are normally smaller tables that grow slowly and can be reused in several use cases. Better suited to being it’s own dimension.



	zip_code	location
0	10506	King's Landing
1	80976	Dorne
2	78956	Braavos
3	67305	Harrenhal
4	25089	Winterfell
5	48732	Karhold
6	35786	Tarth
7	93612	Dragonstone

Table 3: dim_company

- Renamed columns to remove white space and match snake case of other column headers.

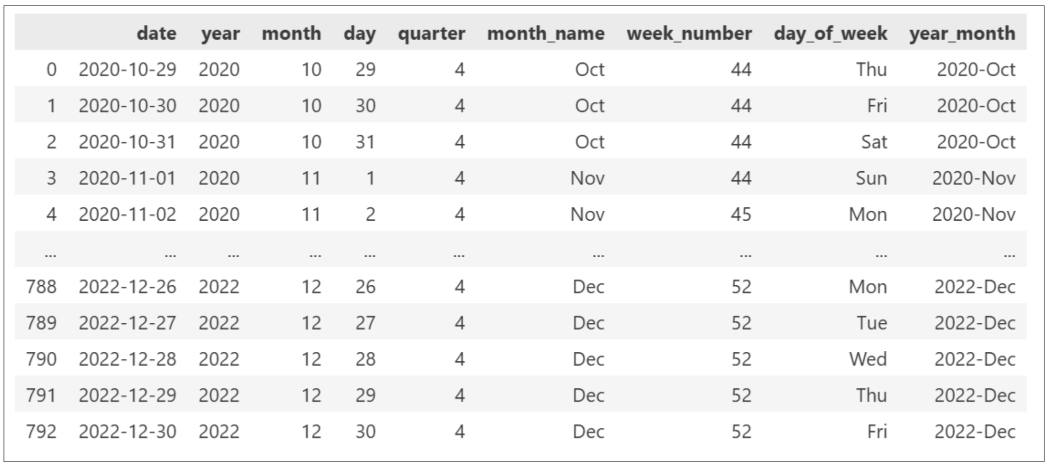


	company_id	company_name	establishment_date	num_employees
0	0	Acme Inc.	2019-12-31	404
1	1	Best Corp.	2021-03-21	229
2	2	Bright Future Enterprises	2020-11-14	222
3	3	Delta Inc.	2020-11-06	662
4	4	Echo Enterprises	2022-05-05	255
5	5	Fast Track Inc.	2020-03-08	374
6	6	Global Enterprises	2022-02-03	896
7	7	High Hopes Inc.	2020-03-17	812
8	8	Infinite Solutions	2021-03-12	436
9	9	Jumpstart Corp.	2020-05-08	872

- establishment_date was converted from a unix timestamp to a date using pandas.

Table 4: dim_date

- A date dimension was created using the minimum and maximum values of the updated_at field as the range. This fills in any gaps in the date data allowing for better visualisations.
- A year_month field was created to enable easier aggregations and visualisations.
- Fields such as year, month, and quarter were created for use in dashboards and reports.

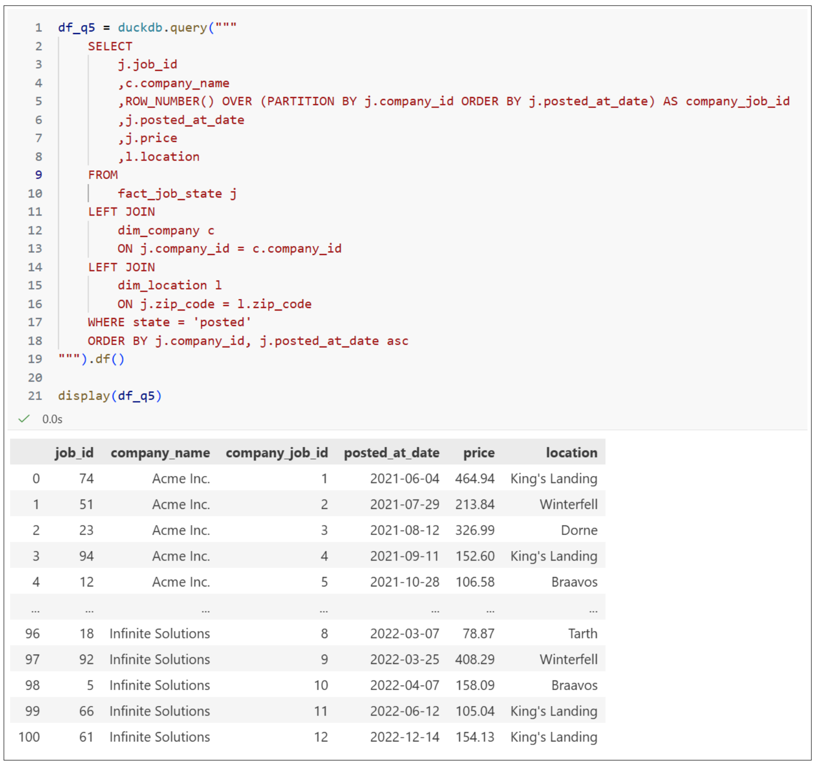


	date	year	month	day	quarter	month_name	week_number	day_of_week	year_month
0	2020-10-29	2020	10	29	4	Oct	44	Thu	2020-Oct
1	2020-10-30	2020	10	30	4	Oct	44	Fri	2020-Oct
2	2020-10-31	2020	10	31	4	Oct	44	Sat	2020-Oct
3	2020-11-01	2020	11	1	4	Nov	44	Sun	2020-Nov
4	2020-11-02	2020	11	2	4	Nov	45	Mon	2020-Nov
...	...	...	...	...	...	...	...	...	...
788	2022-12-26	2022	12	26	4	Dec	52	Mon	2022-Dec
789	2022-12-27	2022	12	27	4	Dec	52	Tue	2022-Dec
790	2022-12-28	2022	12	28	4	Dec	52	Wed	2022-Dec
791	2022-12-29	2022	12	29	4	Dec	52	Thu	2022-Dec
792	2022-12-30	2022	12	30	4	Dec	52	Fri	2022-Dec

- Potential improvements to model design;
  - Add a new dimension for historical job details that would hold previous states for each job. The fact table would change from having multiple rows per job_id to having only one row per job_id. Job dates and states would move to a new dimension table and be joined to the fact table by job_id.

5. Using your dimensional model, write a SQL query that returns a list of jobs for each company, ordered and enumerated within each group by the `posted_at` date.

The query joins the job, company, and location tables and uses a window function to partition the data on company_id, and assign a number to each job in ascending order by posted_at_date.



```
df_q5 = duckdb.query("""
    SELECT
        j.job_id
        ,c.company_name
        ,ROW_NUMBER() OVER (PARTITION BY j.company_id ORDER BY j.posted_at_date) AS company_job_id
        ,j.posted_at_date
        ,j.price
        ,l.location
    FROM
        fact_job_state j
    LEFT JOIN
        dim_company c
        ON j.company_id = c.company_id
    LEFT JOIN
        dim_location l
        ON j.zip_code = l.zip_code
    WHERE state = 'posted'
    ORDER BY j.company_id, j.posted_at_date asc
""").df()

display(df_q5)
```

✓ 0.0s

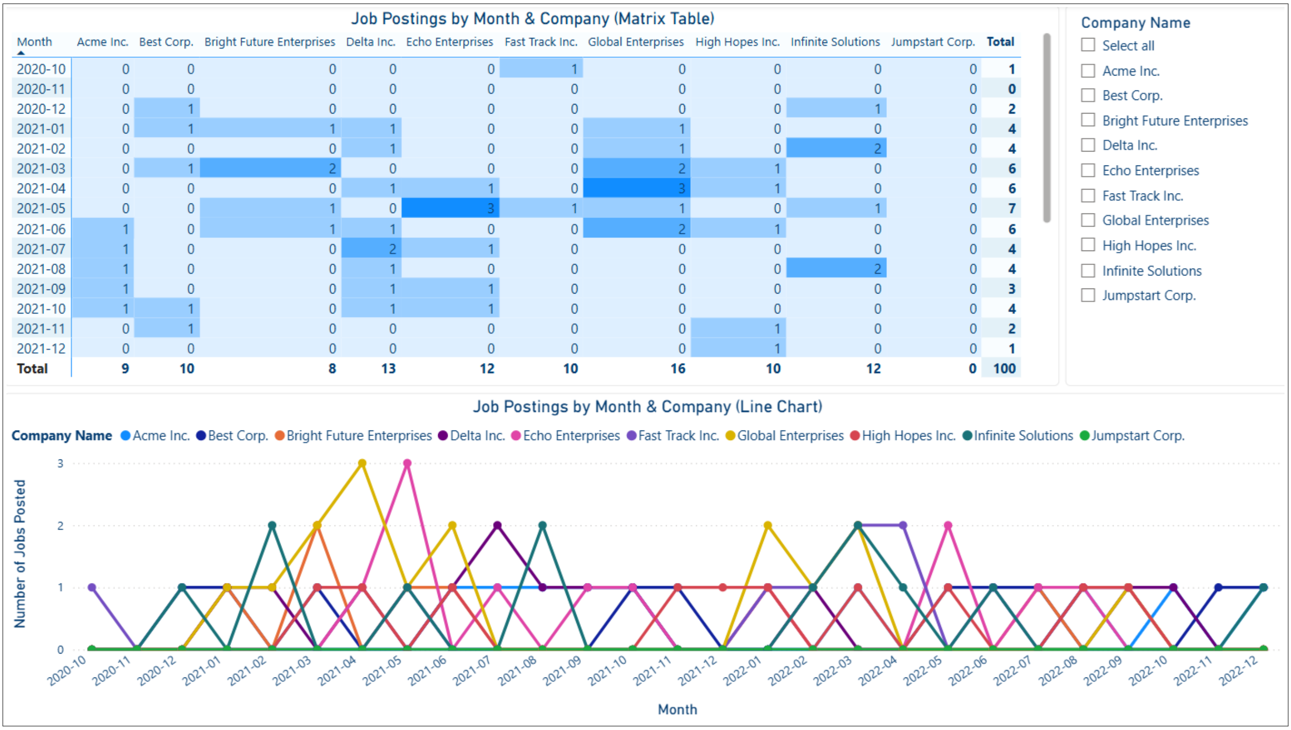
	job_id	company_name	company_job_id	posted_at_date	price	location
0	74	Acme Inc.	1	2021-06-04	464.94	King's Landing
1	51	Acme Inc.	2	2021-07-29	213.84	Winterfell
2	23	Acme Inc.	3	2021-08-12	326.99	Dorne
3	94	Acme Inc.	4	2021-09-11	152.60	King's Landing
4	12	Acme Inc.	5	2021-10-28	106.58	Braavos
...	...	...	...	...	...	...
96	18	Infinite Solutions	8	2022-03-07	78.87	Tarth
97	92	Infinite Solutions	9	2022-03-25	408.29	Winterfell
98	5	Infinite Solutions	10	2022-04-07	158.09	Braavos
99	66	Infinite Solutions	11	2022-06-12	105.04	King's Landing
100	61	Infinite Solutions	12	2022-12-14	154.13	King's Landing

6. Discuss how you would obtain and model information (within your schema) about the duration of jobs (from `posted` to `expired` states).

This can be accomplished with the days_in_state field that was added to the fact_job_state table. The logic for this field calculates how long each job has been in each state. For the current data, filtering to job_ids that only have an expired state, and selecting the days_in_state for posted will provide the number of days between posted and expired.

If jobs had multiple states before reaching expired, using SUM() on the days_in_state field (excluding the days in state for expired) would provide the duration. This would also work if it was possible for jobs to return to posted state after being expired, depending on if it would be considered a new job posting or not.

7. Provide a visualization showing the trend of the number of active jobs over time. Also include the company name data in this visualization.



Job Postings by Month & Company (Matrix Table)

Month	Acme Inc.	Best Corp.	Bright Future Enterprises	Delta Inc.	Echo Enterprises	Fast Track Inc.	Global Enterprises	High Hopes Inc.	Infinite Solutions	Jumpstart Corp.	Total
2020-10	0	0	0	0	0	1	0	0	0	0	1
2020-11	0	0	0	0	0	0	0	0	0	0	0
2020-12	0	1	0	0	0	0	0	0	1	0	2
2021-01	0	1	1	1	0	0	1	0	0	0	4
2021-02	0	0	0	1	0	0	1	0	2	0	4
2021-03	0	1	2	0	0	0	2	1	0	0	6
2021-04	0	0	0	1	1	0	3	1	0	0	6
2021-05	0	0	1	0	3	1	1	0	1	0	7
2021-06	1	0	1	1	0	0	2	1	0	0	6
2021-07	1	0	0	2	1	0	0	0	0	0	4
2021-08	1	0	0	1	0	0	0	0	2	0	4
2021-09	1	0	0	1	1	0	0	0	0	0	3
2021-10	1	1	0	1	1	0	0	0	0	0	4
2021-11	0	1	0	0	0	0	0	1	0	0	2
2021-12	0	0	0	0	0	0	0	1	0	0	1
Total	**9**	**10**	**8**	**13**	**12**	**10**	**16**	**10**	**12**	**0**	**100**

Company Name
Select all
Acme Inc.
Best Corp.
Bright Future Enterprises
Delta Inc.
Echo Enterprises
Fast Track Inc.
Global Enterprises
High Hopes Inc.
Infinite Solutions
Jumpstart Corp.
Job Postings by Month & Company (Line Chart)
Company Name ● Acme Inc. ● Best Corp. ● Bright Future Enterprises ● Delta Inc. ● Echo Enterprises ● Fast Track Inc. ● Global Enterprises ● High Hopes Inc. ● Infinite Solutions ● Jumpstart Corp.
Number of Jobs Posted
Month

The above visuals were created by importing the modelled datasets into Power BI. Both charts are displaying the same information with a slicer to filter by company.

A measure was used to fill in blanks with 0 so companies with no job postings would still be displayed. The visuals are filtered to jobs where state is ‘posted’.

A line chart is commonly used for time series analysis but because there are 9 companies and many of them have a similar amount of jobs posted over time, it is difficult to differentiate each company on the line chart.

On the matrix table it is easier to differentiate between each company and total jobs for each company and month are also displayed. The colour gradient makes identifying trends and anomalies over time possible with a matrix table. The matrix table will also scale better than a line chart for displaying a larger number of companies.