Job Market Analysis

1st Venkata Karthik Patralapati, 2nd Prasad Jayakumar, 3rd Anbu Valluvan Devadasan,
4th Madhulika Dutta, 5th Sai Krishna Bhamidipati
San Jose State University, San Jose, CA,United States of America
Department of Applied Data Science,Data Analytics
DATA 225 Group Project,Group-06

Abstract - The job market might be rather unpredictable, but by properly evaluating the data, we can still learn important lessons. So, in order to better help you grasp the abilities and other crucial information about the Data analyst job posts, we did an analysis. In order to do this, we received a dataset in CSV format from Kaggle that combines recent and historical data. The raw data was then converted utilizing two processing stages: a staging layer and a primary layer. In Google BigQuery, which we're using as a data warehouse, we saved the information in facts and dimensions tables for the main layer. To display our findings, we lastly produced a variety of charts and infographics in Tableau.

I. OBJECTIVE

The aim of our project is to perform cloud analytics and data warehouse implementation for historical data. Following the identification of appropriate entities from the selected dataset, an entity-relationship model is created, and a data warehouse is built to create an online transaction processing (OLTP) database with a star schema model that includes fact tables and other dimension tables. A data pipeline is orchestrated using the Extract, Transform, and Load (ETL) procedure with the addition of analytics through structured querying and visualization on top of it. We thus seek to summarize trends from the selected historical dataset and aid the job-search process of people interested in working in data-oriented roles, making it easier for them to develop their skills for better career prospects and qualify for multiple jobs.

II. METHODOLOGY

A.Dataset Selection

The dataset used in this project is on data analyst job postings and was obtained from the website Kaggle and through SERP API. The Kaggle dataset was collected using web scraping, while the SERP API was used to collect data from Google search results. It is a continually updated historical dataset, starting from November 2022, that stores over 16000 job listings for data analysts in the US from Google's search engine. The data gets updated with over 100 new job posts daily and covers fields such as job location, salary, work schedule, required skills, etc. that are highly relevant for job seekers.

The data source was chosen keeping in mind the industry requirement of expertise in handling constantly updated data, and as such implementing ETL processes on this dataset would help gain practical knowledge of the methodologies. The subsequent analysis performed on the data would also be beneficial for our peers in assessing internship and job prospects for their chosen major.

B.ETL Process

ETL abbreviates to Extract ,Transform and load which basically summarizes the conversion of data from one format to another format where we will be having one source system's data be loaded into another system as a target by applying different kinds of transformations.

In our analysis we have used ETL process where we are taking a csv file from the Kaggle and moreover we are also getting data from API from the google search and we are loading the API data as a json file into the cloud.Here we are considering the csv file as a source system for our ETL process and then we are connecting the Apache Airflow through the Google Big Query and we are reading the data and then we are executing the transformation process into 2 different layers where one is the staging layer and the other would be the main layer.

In the staging layer we are taking the raw data csv and json files as our sources and performing some

data quality as a part of our process as our endpoint is to perform an analysis on the transformed data and we achieve it by taking the staging layer targets as our sources and then we split them into different dimensions and fact tables and store them in our Data warehouse.

Reason for choosing ETL

Even though we have our cloud storage ven for the source we have used ETL over ELT for two main reasons one is we are using staging layer for which primary goal is to perform the data quality check.Secondly our end point is to perform analytics on the transformed data.So to attain these measures ETL processes would be a great choice when compared with ELT.

C. Cloud Architecture

In today's data driven world, processing of data is a crucial process which is a core process of any company that deals with or without data.

The objective of the project is to explain the significance of cloud based ETL implementation using Google Cloud and its services. The project also includes the benefits of using a cloud-based ETL to improve the scalability, performance and reliability of the data processing.

Background:

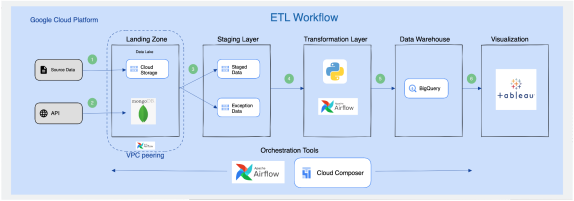
ETL (Extract, Load and Transform) is the process of extracting data from multiple sources of an organization or a project, transforming it into a usable format for data analysis and finally loaded to a target system for analysis. This part of the system is critical to the entire data analysis process which is crucial in getting clean and trustworthy data about the business. The traditional on-premise ETL systems use the system to extract data from different sources or applications and then load it to a designated data warehouse or data-lake for analytics.

However, the traditional ETL approach has its own limitations that include performance, scalability and maintenance issues and high costs. Thus cloud computing becomes a handy and trustable process that bridges the gap found in traditional ETL systems. Especially when it comes to performance, scalability and maintenance, cloud becomes the go to solution these days.

Several cloud providers have their own set of tools and the number of tools to achieve the ETL is increasing rapidly. However, as part of the project, we have used Google Cloud Platform as the suite of tools provided are user friendly and have seamless integrations.

Architecture:

The cloud based ETL setup using google cloud that we have come up with includes, Cloud Composer which is our orchestrator tool that is built on top of Apache Airflow, and storages as Cloud storage buckets as well Big Query as our data warehouse. The architecture of the cloud base ETL pipeline is shown in the diagram below.



Architecture diagram 1: ETL workflow

The design consists of following components:

1. ***Data Sources*:** There are two types of data sources that we use, one is CSV source from Kaggle which is a daily updated resource and other is an API from SERP API.
   Additionally, we wanted to mock the source as NoSQL for the project, we load the data from API which is a json formatted one, to the Cloud MongoDB atlas.
2. ***Cloud Storage:*** The data extracted from multiple sources are landed onto a google cloud storage bucket which acts as our landing zone. This is a highly available and durable storage object service provided by the cloud platform. In addition, we also have MongoDB as our data storage which is a cloud instance of MongoAtlas that is again one of the widely used NoSQL cloud storage.

3. ***Cloud Composer:*** It is a fully managed workflow orchestration tool that automates the entire ETL process that helps in providing specific environments for managing, testing and maintaining different ETL workflows on cloud.
4. ***Airflow:*** It is an open-source platform that provides necessary tools for achieving the pipeline for ETL. It consists of a Web UI interface to manage the workflows, a scheduler for executing the pipeline and worker nodes for performing the tasks.
5. ***Bigquery:*** This is the data warehouse of the pipeline, where all the transformed data are loaded into for further analysis. With the efficient NoSQL backend and SQL interface, the data warehouse plays a crucial role in the ETL operations.
6. ***Tableau:*** The visualization tool that we included in the project is Tableau which is a licensed version that is easily integrated with the Big Query for the visualizations. The sheets and dashboard features provide a great platform for better data reporting.

Implementation:

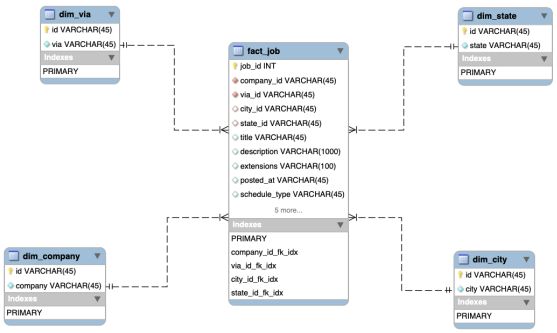
Following are the steps for the implementation of the ETL pipeline based on the architecture discussed above.

1. Create a cloud storage bucket for storing the extracted data from various sources. This can be achieved by using google cloud platform console UI or using command line interface, but with a google cloud account. This is part of the landing zone which is the layer where all the data from different sources are landed initially.
2. Create another set of cloud storage buckets that represents the staging layer, where all the data collected from different sources are merged and then scanned for dirty data to be loaded into exception buckets.
3. Create a cloud composer environment and configure with the necessary dependencies for the transformation scripts. This will in turn create containers for Airflow instances and storages that are needed and this is part of the orchestration layer of the project.
4. Create an Airflow DAG (Directed Acyclic Graph) that has the definition of the pipeline which also includes the transformation scripts that are necessary for data cleaning and transformations. Create environment variables for storing critical information and setup schedulers for recurring and scheduled pipelines.
5. Define various tasks in the Airflow DAGs that includes the gathering of data from sources to landing zone, merging data into a common staged data, task for scanning dirty records as well as transformation and loading data to the target warehouse system. The tasks can be defined using Python and SQL operators in the DAG.
6. Establishing a connection to Bigquery is achieved by creating Google Cloud Bigquery operator available in python.
7. Create a new licensed version of Tableau instance where the location of the server can be targeted to Big Query using Oauth authentication. Tableau sheets and dashboards can be built based on the business requirements.
8. Run the DAG pipeline, schedule and monitor the activity through the Cloud Composer, Airflow UIs and also through the logs. The Composer UI provides a dashboard for monitoring the statuses of the environment while the Airflow UI gives the in detail activity of the workflow pipelines.

Bigquery:

Bigquery is a cloud based data warehouse service and business intelligence tool developed by Google that is designed to store and analyze large amounts of data quickly and efficiently. The reason for utilizing this service is because of its easily scalable, speed, cost-effective and also integrated with various google cloud products and services along with visualization tools like Tableau.

As BigQuery is flexible and we wanted to leverage the start schema for handling our data in the data warehouse, we came up with a fact table and four dimension tables to store and process the data.
The schema of the Bigquery tables for our project is shown in the below figure.



BigQuery star schema

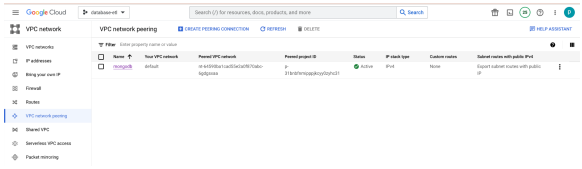
MongoDB using VPC peering:

While using MongoDB cloud instances with ETL tools in the cloud, it is always advisable to use VPC peering to secure the connection and efficient data transfer.

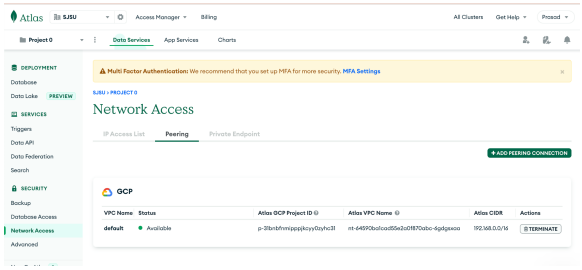
VPC (Virtual Private Cloud) peering is a method to connect to VPC together to facilitate communication between instances in the VPC to connect securely without going through public internet.

Why is it important?

1. Security: while transferring sensitive data such as financial information or personal information through the cloud, it is important to make sure the data is secure and hence VPC plays a crucial role in ensuring data security.
2. Efficient data transfer:
   While considering the data transmitted to be fast and efficient, VPC peering provides the networking platform and connections without network congestion or latency issues.
3. Cost savings:
   Using VPC peering can lead to cost savings as public internet usage may incur more cost in the long run.



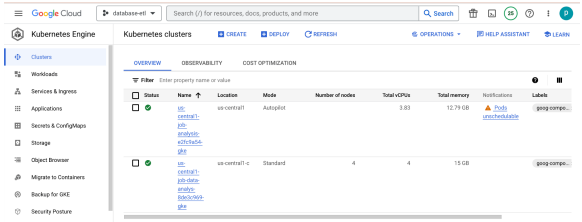
VPC peering in google cloud to MongoDB atlas



VPC peering in MongoDB atlas to enable two way communication from Google Cloud

Autoscaling of cloud composer:

The cloud composer instance that we created in google cloud is version 1 which doesn't facilitate autoscaling. We tried creating composer version 2 which facilitates the autoscaling of the worker nodes, however with the resources available, we were unable to successfully create the composer version 2. Hence in order to leverage the usage of Airflow and considering the data load we used the available free resource of composer version 1.



Cloud composer version 2 diagram with autoscaling

D.Airflow Pipeline

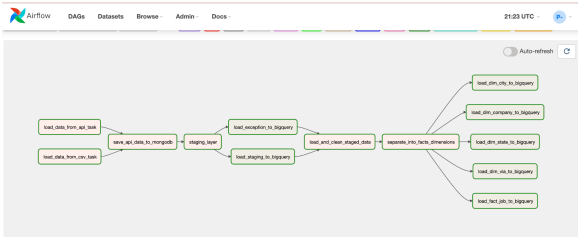
Airflow is an open source platform for creating, scheduling and managing data workflows. Initially designed and developed for Airbnb, in recent years, it has gained popularity by widely being adopted by

various organizations for managing the data workflows. As part of the cloud adoption for ETL processes, Airflow is a crucial tool that helps in a variety of aspects in terms of scalability, performance and monitoring.

The architecture of Airflow consists of the following components:

1. ***Scheduler:*** The scheduler is responsible for scheduling the tasks in the workflow. Based on the DAG definitions, the tasks are executed as per the configured schedules.
2. ***Workers****:* The workers are responsible for executing the tasks that are defined in the DAGs. Each task is executed by a separate worker and they can be scaled up and down based on the needs.
3. ***Metadata Database:*** The metadata database is used to keep track of the tasks being executed. It is also used by the scheduler to track the tasks that are completed and the ones that are yet to run.
4. ***Message Broker:*** The message broker is responsible for passing the messages between components and also the execution of tasks. They make sure that each task is executed once in a pipeline run.
5. ***Executors:*** They are responsible for the task execution and there are several executors available in Airflow, such as LocalExecutor, SequentialExecutor, CeleryExecutor, and KubernetesExecutor. The choice of executor depends on the workload and the infrastructure.

The workflow of the pipeline of the project is depicted in the below diagram.



Airflow Pipeline Diagram

The following are the steps of the pipeline execution:

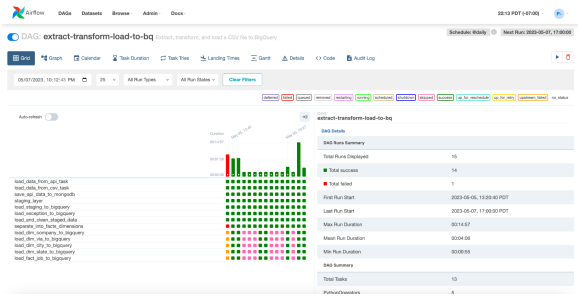
1. Load data from api: This task extracts the data from the api and loads it into a storage bucket. This is ideally to mock the transactional data store where the data is stored in the noSQl database later.
2. Load data from csv: The other source of our data is from csv, where this task extracts data from the csv and loaded into bucket storage which is the landing zone.
3. Load data to MongoDB: As a source of transactional database, all the api data which is a json format is loaded into Cloud MongoDB Atlas instance. This task makes sure all the data is loaded into the noSQL database from the api.
4. Staging: This task is responsible for merging data from all data sources and then adding to a single table. It also separates the dirty records to exception tables
5. Load exception to Bigquery: The task takes care of loading the exception data to Bigquery tables for analysis.
6. Load staged data to Bigquery: The tasks load the merged/cleaned data into Bigquery tables for analysis.
7. Load and clean staged data: The data is then cleaned and transformed as needed and stored into intermediate storage.
8. Separate into Facts and Dimensions : The task will load the cleaned and transformed data and separate them into facts and dimensions.
9. Load into facts and dimensions: This final task will load all the separated facts and dimensions data into Bigquery tables for further analysis.

Airflow features:

We explored and adopted various Airflow features in our project.

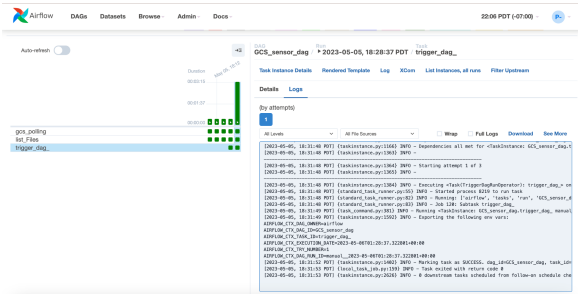
1. **Scheduler (daily):**

   We scheduled our ETL pipelines to run daily at a fixed time. This enabled us to extract and load data daily leveraging the advantage of our daily updating dataset.



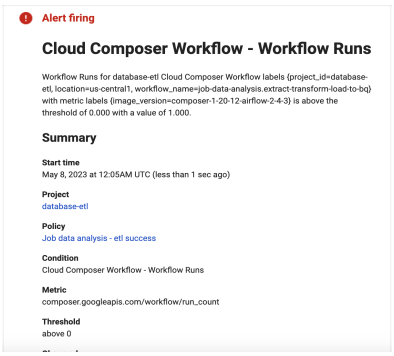
2. **Gcs update sensor trigger: (GoogleCloudStorageObjectUpdatedSensor)**

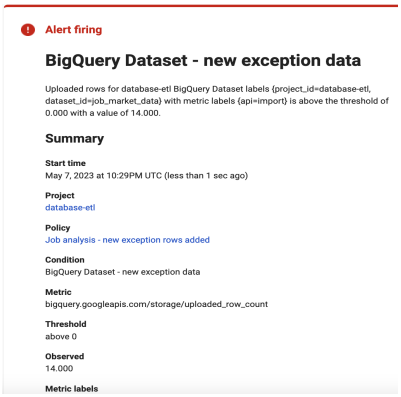
The sensor library available in python and google cloud that enables us to sense newly arrived or updated files in the storage/landing area and trigger the pipeline run of our ETL process.



3. **Mail alerts(success, failure and exceptions)**

The monitoring feature of the cloud platform enabled us to embed the mail alerting feature by creating rules and alert policies. We created one for each success and failure runs of the ETL pipelines. In addition, we also explored the functionality of alerting whenever there is dirty records more than a threshold value which enable the business to alert mishap and take immediate action.





E. Analysis

The primary objective of the data analytics process is to extract relevant information for deriving conclusions from the data. Our dataset has over 15000 entries for job postings till date which would be difficult to go through and interpret individually. We are thus leveraging database querying and visualization techniques to generate meaningful insights from the data. The analysis was carried out using SQL queries on the Google Big Query platform and the tables obtained were further used for building reporting dashboards in Tableau. The queries performed and results generated are detailed in the following sections.

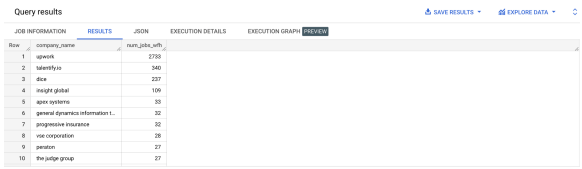
i) Big Query (SQL) Queries and Results:

1. Number of Work from home (WFH) options by each company.

Query:

```
SELECT
  c.company_name,
  COUNT(*) AS num_jobs_wfh
FROM
  job_market_data.fact_job j
JOIN
  job_market_data.dim_company c
ON
  j.company_id = c.id
WHERE
  j.wfh = 'True'
GROUP BY
  c.company_name
ORDER BY
  num_jobs_wfh DESC
```

Result:

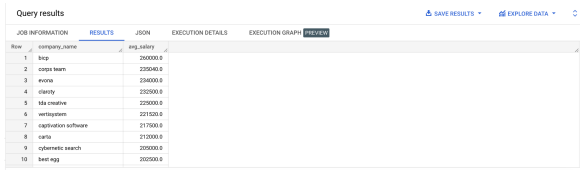


2. Average Salary by Top companies.

Query:

```
SELECT
 c.company_name,
 AVG(j.salary) AS avg_salary
FROM
 job_market_data.fact_job j
JOIN
 job_market_data.dim_company c
ON
 j.company_id = c.id
GROUP BY
 c.company_name
ORDER BY
 avg_salary DESC
```

Result:

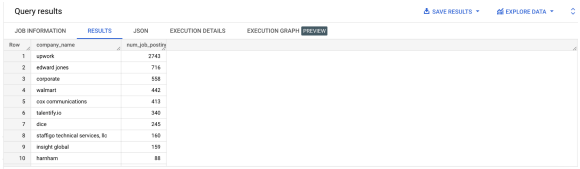


3. Count of the number of job postings by each company.

Query:

```
SELECT
  c.company_name,
  COUNT(*) AS num_job_postings
FROM
  job_market_data.fact_job j
JOIN
  job_market_data.dim_company c
ON
  j.company_id = c.id
GROUP BY
  c.company_name
ORDER BY
  num_job_postings DESC
```

Result:



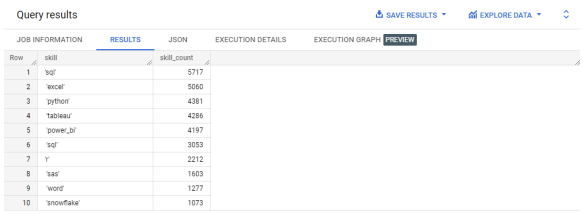
4. Identifying the most mentioned skill from a variety of job postings.

Query:

```
SELECT
  skill,
  COUNT(*) AS skill_count
FROM (
  SELECT
    c.company_name,
    f.title,
    STRUCT(
      ARRAY(
                SELECT  REPLACE(REPLACE(SPLIT(skills,  ',
')[OFFSET(0)], '[', ''), ']', '') AS skill
        FROM UNNEST(SPLIT(skills, ',')) AS skills
      ) AS skills
    ) AS skills_list,
    v.via,
    state.state,
    city.city
  FROM
    `database-etl.job_market_data.fact_job` f
  JOIN
              `job_market_data.dim_company`  c  ON
f.company_id = c.id
  JOIN
    `job_market_data.dim_via` v ON f.via_id = v.id
  JOIN
       `job_market_data.dim_city`  city  ON  f.city_id  =
city.id
  JOIN
    `job_market_data.dim_state` state ON f.state_id =
state.id
  WHERE
```

```
    f.skills IS NOT NULL
)
CROSS JOIN
  UNNEST(skills_list.skills) AS skill
WHERE
  skill != ''
GROUP BY
  skill
ORDER BY
  skill_count DESC
LIMIT
  10
```

Result:

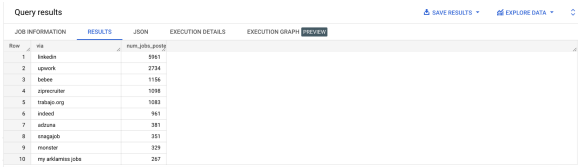


5. Top 10 Sources and no of jobs posted via/ through them.

Query:

```
SELECT
  via,
  COUNT(*) AS num_jobs_posted
FROM
  job_market_data.fact_job j
JOIN
  job_market_data.dim_via v ON j.via_id = v.id
GROUP BY
  via
ORDER BY
  num_jobs_posted DESC
LIMIT
  10;
```

Result:

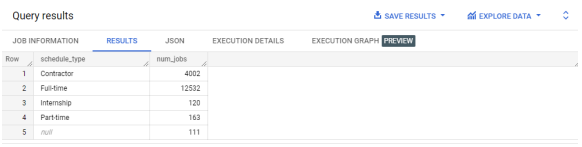


6. Total no of jobs that are being offered based on scheduled_type.

Query:

```
SELECT schedule_type, COUNT(*) AS num_jobs
FROM job_market_data.fact_job
GROUP BY schedule_type
```

Result:

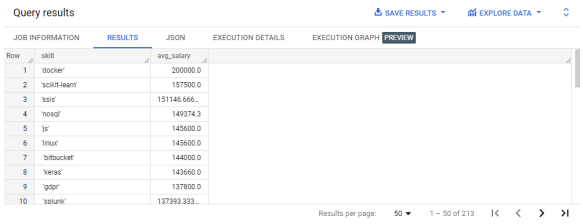


7. Avg_salary for distinct skill.

Query:

```
 SELECT
  skill,
  AVG(salary) AS avg_salary
FROM (
  SELECT
    STRUCT(
      ARRAY(
                SELECT  REPLACE(REPLACE(SPLIT(skills,  ',
')[OFFSET(0)], '[', ''), ']', '') AS skill
        FROM UNNEST(SPLIT(skills, ',')) AS skills
      ) AS skills
    ) AS skills_list,
    salary
  FROM
    job_market_data.fact_job
  WHERE
    skills IS NOT NULL AND salary > 0
)
CROSS JOIN
  UNNEST(skills_list.skills) AS skill
GROUP BY
  skill
ORDER BY
  avg_salary DESC;
```

Result:

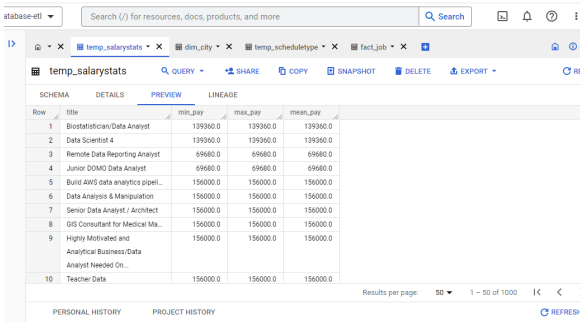


8. Salary statistics by job title.

Query:

```
SELECT title,
    MIN(salary) AS min_pay,
    MAX(salary) AS max_pay,
    AVG(salary) AS mean_pay
FROM database-etl.job_market_data.fact_job
WHERE salary > 0
GROUP BY title
LIMIT 1000;
```

Result:



9. Job Postings and mean salary by job schedule.

Query:

```
SELECT schedule_type,
    COUNT(schedule_type) AS job_postings,
    AVG(salary) AS avg_pay
FROM database-etl.job_market_data.fact_job
WHERE schedule_type != ''
GROUP BY schedule_type
ORDER BY job_postings DESC;
```

Result:

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAP

Row	schedule_type	job_postings	avg_pay
1	Full-time	13267	99236.7450...
2	Contractor	4307	92629.0985...
3	Part-time	172	91552.6666...
4	Internship	123	47840.0

10. Top 10 Cities with Highest Avg Salary.

Query:

```
SELECT city,
    AVG(salary) AS avg_pay
FROM job_market_data.fact_job c
JOIN job_market_data.dim_city s ON s.id = c.city_id
WHERE salary > 0
GROUP BY city
ORDER BY avg_pay DESC;
```

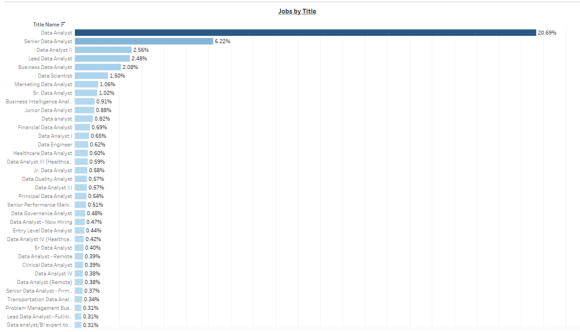
Result:

SCHEMA | DETAILS | PREVIEW | LINEAGE

Row	city	avg_pay
1	san francisco	155679.333...
2	cupertino	153920.0
3	eugene	144481.5
4	tebbetts	144481.5
5	meta	144481.5
6	fulton	144481.5
7	california	144481.5
8	lake lotawana	144481.5
9	loose creek	144481.5
10	st robert	144481.5

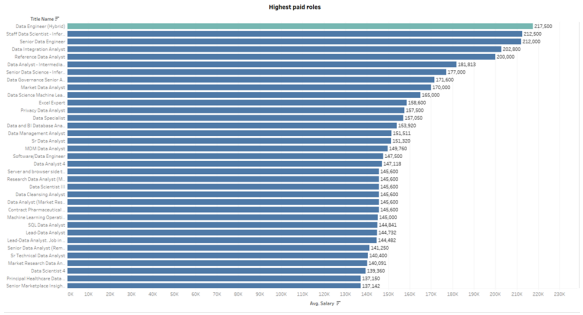
ii) KPIs and Visualization Results:

1. Number of Job postings: This KPI would provide the overall demand for the data analyst in the job market and also the level at which there is maximum demand.



There is huge demand for Data Analyst at the entry level as per the data available with us.

2. Salary range: This would give an insight into the salary expectations of the employers and the compensation level being offered in the market.

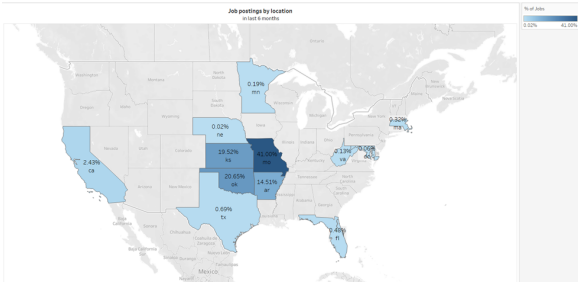


Data Engineer earns the highest salary of $217,500 and as it is evident the average salary for majority of the data analyst roles are in 6 figures.

3. Required Skills: This KPI would give us an idea of the most in-demand skills at the industry level for data analyst roles and could help us identify any gaps in skills and training.

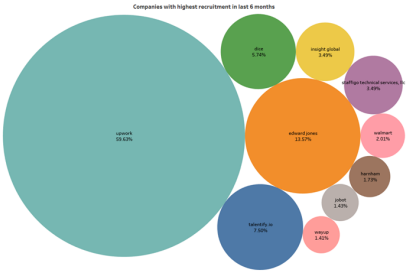
Most in demand skills for data analyst roles are SQL, Power_BI, Tableau, Python and excel among others.

4. Job location: This would give us insight into the geographical distribution of the data analyst jobs and identify regional differences in demand, salaries and required skills.



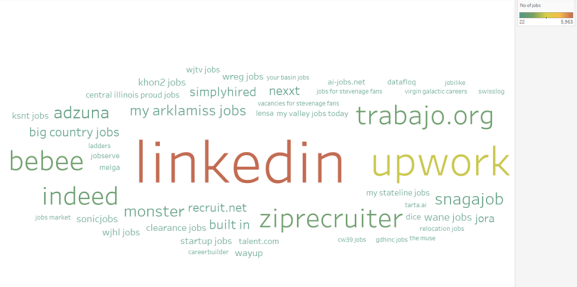
Surprisingly, there is a huge demand for the data analyst roles in central american states and close to 95% of vacancies posted in the last 6 months have come from 4 central states of Missouri, Kansas, Oklahoma and Arkansas.

5. Company size: This KPI would give us information about the size of companies that are hiring data analysts and how it affects job requirements, salaries and other factors.



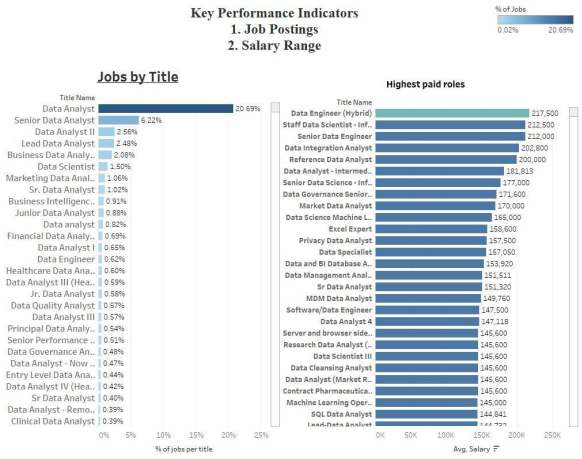
Top 5 companies with highest recruitment in the last 6 months have been upwork, edward jones, talentify.io, dice and insight global.

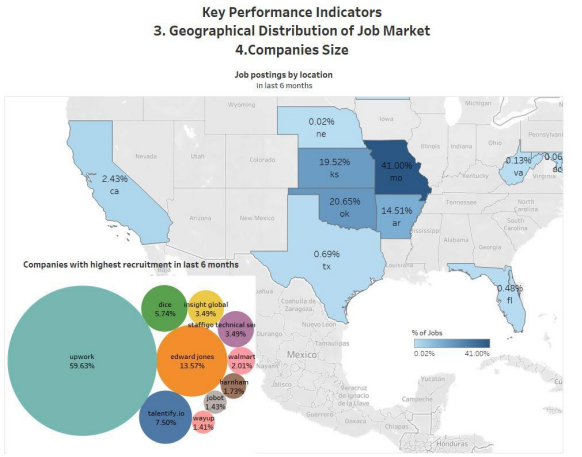
6. Recruitment Channels: This would give us a brief idea of the most effective channels for recruiting data analysts, such as job boards, social media, referrals etc.



The most preferred recruitment channels have been Linkedin, upwork, Indeed, Bebee amongst few others.

DashBoard





III. CONCLUSION

Our project demonstrates the use of cloud analytics and data warehousing for dealing with historical data. Due to the continuous update feature and large size of our dataset, we have utilized workflow management tools such as Google Cloud Composer and Airflow for building our data pipeline. This ensured that most of the ETL processes were automated to run as scheduled leading to significant time savings and highly reduced manual effort.

The choice of Google Cloud Platform for deploying our project provided secure and scalable cloud storage for the data warehouse along with easy retrieval for further analysis. By querying the database through Google Big Query we were able to generate insights into the data and leveraged them for building visualizations on Tableau. Starting with a large collection of job-related data, we were thus able to successfully summarize it as reproducible and meaningful information by leveraging analytical tools and processes.

IV. REFERENCES

1. Kaggle data set Data Analyst Job Postings [Pay, Skills, Benefits] https://www.kaggle.com/datasets/lukebarousse/data-analyst-job-postings-google-search
2. SerpAPI - Google Jobs https://serpapi.com/search?engine=google_jobs
3. Sample get api https://serpapi.com/search.json?engine=google_jobs&q=barista+new+york&hl=en