Some assumptions I made in analyzing this data include:

1. ‘Event Occurred At’ column represents time stamp for event the job was assigned and ‘Request Created At’ is when the client submitted a job request on the website
2. ‘Quality Score Scoring and Quality Score Writing’ are existing scores for each analyst when an event is created, and not the score on the requested job
3. From timestamp of ‘Event Occurred At’, sequence of action steps = Assigned job then either accepted or declined

PART 1 : ANALYSIS

I began my analysis with an exploration of the number of distinct requests in the dataset (74), took a look at the distribution of the requests by examining requests per day - over the three day stretch, 47% of requests came on 6/22/17 - 35, and requests per hour - 19% of requests were at 12 UTC. The preliminary insights from the exploratory analysis can be used to make hiring and staffing decisions of analysts, preferably investigation analysts (or perhaps assigners) availability alongside request data to ensure that analysts are available at peak request time.

Requests per Day

```
SELECT
  TIMESTAMP_TRUNC(request_created_at,
    DAY) day,
  COUNT(DISTINCT request) as request_count
FROM
  `take_home.assignment_log`
GROUP BY
  1
ORDER BY
  1 DESC
```

Requests per Hour

```
SELECT EXTRACT(HOUR FROM request_created_at) AS hr_requested,
COUNT (DISTINCT request) AS num_events
FROM `take_home.assignment_log`
GROUP BY hr_requested
ORDER BY num_events DESC
```

Knowing the general distribution of the requests, I then jumped into analyzing the analysts data as the bulk of business spend is on human capital and the large distributed workforce is at the

heart of Wonder. Some interesting insights here include discovering a ~90% conversion rate between analysts and # of times they accept the jobs they are assigned. This speaks well of the business at a micro scale - that a subset of the pool of analysts are highly engaged. A larger sample size would be needed to test to see if this hypothesis is true. Finally, in the attached table, I investigate more deeply the 10% declines by analysts in the sample data. Here, we can read the count of declines per analyst per job (tells us that there is some redundancy in assignment even after first decline), what types of jobs analysts are declining, their wait times, what they are waiting for, and total jobs available. I can infer from the query results that long wait times coupled with job availability, and multiple reasons for waiting leads to a declined job.

Analysts Assignment Conversion Rate

```
WITH analysts AS(
SELECT
COUNT(DISTINCT analyst) AS total_analysts
FROM take_home.assignment_log
),

accepts AS(
SELECT
COUNT(DISTINCT analyst) AS total_accepts
FROM take_home.assignment_log
WHERE Action = 'accepted Job'
)

SELECT
  total_analysts,
  total_accepts,
  total_accepts/ total_analysts AS conversion_rate
FROM analysts, accepts
```

Row	total_analysts	total_accepts	conversion_rate
1	71	64	0.9014084507042254

Analysts Decline And Potential Factors

```
SELECT
DISTINCT(analyst),
request,
COUNT(Distinct Event_occurred_at) AS count_declined,
```

```
job,
wait_time__min_,
waiting_for,
total_jobs_available
FROM take_home.assignment_log
WHERE action = 'Declined Job'
GROUP BY analyst, request, job, wait_time__min_, waiting_for, total_jobs_available
ORDER BY count_declined DESC
```

PART 2 : DATA MODELING

To model this data in a warehouse, I created a table with abbreviations of the field names for use in the model. The goal was to normalize the data by removing partial and transitive dependencies as seen in the tables created below.

Field name	Type	Mode	ABBR.
Event_occurred_at	TIMESTAMP	NULLABLE	EOA
Analyst	STRING	NULLABLE	ANA
Quality_score__sourcing_	FLOAT	NULLABLE	QSS
Quality_score__writing_	FLOAT	NULLABLE	QSW
Action	STRING	NULLABLE	ACT
Request	STRING	NULLABLE	REQ
Request_created_at	TIMESTAMP	NULLABLE	RCA
Job	STRING	NULLABLE	JOB

Wait_time__min_	INTEGER	NULLABLE	WTM
Waiting_for	STRING	NULLABLE	WF
Analysts_available	INTEGER	NULLABLE	AA
Analysts_occupied	INTEGER	NULLABLE	AO
Total_jobs_available	INTEGER	NULLABLE	TJA
Review_jobs_available	INTEGER	NULLABLE	RJA
Vetting_jobs_available	INTEGER	NULLABLE	VJA
Planning_jobs_available	INTEGER	NULLABLE	PJA
Editing_jobs_available	INTEGER	NULLABLE	EJA
Sourcing_jobs_available	INTEGER	NULLABLE	SJA
Writing_jobs_available	INTEGER	NULLABLE	WJA

EOA	ANA	QSS	QSW	ACT	REQ	RCA	JOB	WTM	WF	AA	AO	TJA	RJA	VJA	PJA	EJA	SJA	WJA

Event to request Table

EOA	REQ	RCA

Analyst to job Table

ANA	JOB

Analyst to action Table

ANA	ACT

Request to job Table

REQ	RCA	JOB

Quality of analyst Table

ANA	JOB	QSS	QSQ

Wait per request Table

REQ	WTM	WF

Analyst availability Table

REQ	AA	AO

Job availability Table

ANA	TJA	RJA	VJA	PJA	EJA	SJA	WJA

3 questions business users might ask:

1. What are the average wait times for each job type?

```
SELECT request_to_job.JOB,
AVG (wait_per_request.WTM)
FROM request_to_job
LEFT JOIN wait_per_request
ON request_to_job.REQ = wait_per_request.REQ
GROUP BY request_to_job.JOB
ORDER BY DESC
```

2. What is the average total job availability for the different actions taken on a job?

```
SELECT analyst_to_action.ACT,
AVG (job_availability.TJA)
FROM analyst_to_action
LEFT JOIN job_availability
```

```
ON analyst_to_action.ANA = job_availability.ANA
GROUP BY analyst_to_action.ACT
```

3. What is the average quality score for sourcing and writing jobs?

```
SELECT quality_of_analyst.JOB,
AVG (quality_of_analyst.QSS),
AVG (quality_of_analyst.QSS)
FROM quality_of_analyst
WHERE quality_of_analyst.JOB = ‘sourcing’ or quality_of_analyst.JOB = ‘writing’
GROUP BY quality_of_analyst.JOB
```

PART 3 : SQL

3. Assuming the product manager intends to get a summary table with each customer's 2009 order amount, below is the fixed query:

```
SELECT
customers.customer_name,
COALESCE(SUM(orders.order_amt), 0) AS total_2009
FROM
customers
LEFT JOIN orders
ON (
customers.customer_nbr = orders.customer_nbr
)
and orders.order_date >= '20090101'
GROUP BY
customers.customer_name
```

With Output:

customer_name	total_2009
Jeff Gordon	0
Jim Brown	48

Julie Peters	0
Peter Green	10.25

Where the issues were:

1. Missing customer names:
   Culprit: Line 9 WHERE vs. And where the WHERE statement only returns rows in the customers table where there is a match on the where clause, and not all the rows from the customers table. Since order_date is only in the orders table, must add that condition to the LEFT JOIN statement with an ‘and’

2. Syntax: SUM(COALESCE) vs. COALESCE(SUM) where the latter performs faster by calling COALESCE one time without processing the function for every record but for a roll in all values are NULL.