Time limit: 5 hours

Please read the entire document before starting. Results should be submitted in the form of an R Notebook (.Rmd file) with the compiled html file also attached. Email the results to [seth.neel@welligence](about:blank) and [akash.sadashivapeth@welligence.com](mailto:akash.sadashivapeth@welligence.com) with subject line:

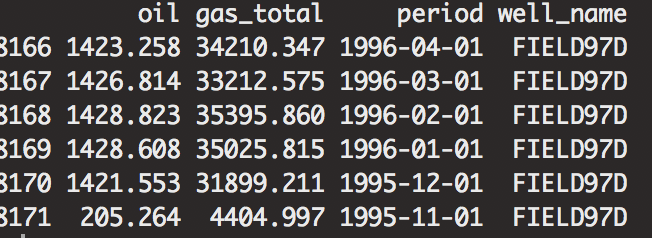
[Your Name]: Completed Technical Interview [Current Date].

Input: test.csv

* Columns: oil, gas, period, well\_name

Each row of the data frame represents a month of production for a specific well in the field.

For example the last row below indicates that well FIELD97D produced 205.264 barrels/d of oil in the month of November in 1995, and 4404.97 m3/d of gas.



**Data Cleaning and Exploration:** *tidyverse/dplyr*

1. produce a “series” data frame where the columns are well names and the rows are the date (period)
2. Create a data frame of well characteristics, where for each well in the data frame

The average gas to oil ratio is calculated, the number of months of production, the

Initial (first month date) of oil production, and the average month over month decline rate for

Gas.

**Forecasting**

* Filter out wells with < 24 months of production, filter out zero production.
* Using the filtered data, build and validate a time series forecasting model that predicts oil production. The time horizon for the prediction is 6 months out into the future. After trying a number of approaches, present and justify the best model. Note: In this part of the interview you are being asked to fit a model using all of the available well data with > 24 months of production, not to forecast a single well time series.

The solution to this part should include both the R code, and a paragraph detailing the methods tried, including any exploratory data analysis or modelling choices, and summarizing your findings.

**Visualization**

Write a function that takes as an input a well name, and outputs a plot of the well historical oil production values, along with the forecasts from your model (in a different color), where the x-axis is months since start of production.

**PostgreSQL**

Given the below database schema, write SQL which outputs a table with the following columns:

asset\_name (this is the name field in the assets table)

basin (this is the basin name from the basins table that corresponds to the asset)

pdp\_oil (from the asset\_reserves table)

average\_oil\_production (calculated as the average of the oil for each asset from the asset\_total\_productions table)

order the table by the value of the average\_oil\_production

A close up of text on a white background

Description automatically generated

**Answer:**

SELECT a.name AS asset\_name,

b.name AS basin,

ar.pdp\_oil,

AVG(atp.oil) AS average\_oil\_production

FROM assets a

LEFT JOIN asset\_reserves ar ON a.id = ar.asset\_id

LEFT JOIN basins b ON a.basin\_id = b.id

LEFT JOIN asset\_total\_productions atp ON a.id = atp.asset\_id

GROUP BY a.id, a.name, b.name, ar.pdp\_oil

ORDER BY average\_oil\_production DESC;

This query uses several joins to combine data from different tables:

LEFT JOIN assets a: This joins the assets table as the main table (a).

LEFT JOIN asset\_reserves ar ON a.id = ar.asset\_id: This joins the asset\_reserves table on the condition that the id of the asset in the assets table matches the asset\_id in the asset\_reserves table.

LEFT JOIN basins b ON a.basin\_id = b.id: This joins the basins table on the condition that the basin\_id of the asset in the assets table matches the id in the basins table.

LEFT JOIN asset\_total\_productions atp ON a.id = atp.asset\_id: This joins the asset\_total\_productions table on the condition that the id of the asset in the assets table matches the asset\_id in the asset\_total\_productions table.

Explanation of the columns:

a.name AS asset\_name: Selects the name field from the assets table and renames it to asset\_name.

b.name AS basin: Selects the name field from the basins table and renames it to basin.

ar.pdp\_oil: Selects the pdp\_oil field from the asset\_reserves table.

AVG(atp.oil) AS average\_oil\_production: Calculates the average of the oil field from the asset\_total\_productions table grouped by asset (a.id) and renames it to average\_oil\_production.

Filtering and Ordering:

GROUP BY a.id, a.name, b.name, ar.pdp\_oil: Groups the results by these columns to ensure we have unique entries for each asset with its corresponding details.

ORDER BY average\_oil\_production DESC: Orders the results by the average\_oil\_production in descending order (highest to lowest).

This query should return a table with the desired columns containing asset information, basin name, PDP oil, and the average oil production for each asset, ordered by the average oil production.

**Other R Experience**

Do you have any experience building RShiny apps? If so please detail it below.

*Yes. I built several simple apps using RShiny apps before.*

*Some examples:*

1. *A basic stock price forecasting app*
2. *Drilling fluid hydraulics app*

*My main dashboarding experience is using streamlit with Python, where I have built some advanced applications (ie apps showing results from an ML model, auto pulling/pushing data between databases, etc)*

Do you have experience collaborating on github?

*Yes.*

What experience if any do you have deploying R in production?

*No major experience with R deployment*

*I have >5 years of experience deploying Python in production, mainly using Apache-airflow, Docker and Azure.*