Technical Questionnaire – Data Analyst (Data Science Competence)

Instructions: Please complete the questions below. Include code snippets in Python/SQL where applicable and clearly explain your approach.

Section A: SQL & Exploratory Data Analysis

1. **SQL Task**:
   You have a table `events` with:

   `user_id` (STRING)
   `event_type` (STRING)
   `event_timestamp` (TIMESTAMP)

Write a query to find the number of unique users who triggered both a `signup` and a `purchase` event within 7 days.

```sql
with signups as
(
   select user_id, min(event_timestamp) as signup_time
   from events
   where event_type = 'signup'
   group by user_id
),
purchases as
(
   select user_id, event_timestamp as purchase_time
   from events
   where event_type = 'purchase'
)
select count(distinct p.user_id) as event_within_7_days
from signups s
join purchases p
on s.user_id = p.user_id
and p.purchase_time between s.signup_time
and timestamp_add(s.signup_time, INTERVAL 7 day);
```

Explanation:
To solve this, I follow these steps:

1. Get the first signup date for each user by filtering for signup events and taking the earliest timestamp per user.

2. Get all purchase events from the same table.

3. Join both datasets on user_id, and filter for purchases that happened within 7 days after the signup date.

4. Count the distinct user IDs who satisfied both conditions.

2. Exploratory Analysis:
Given a dataset with customer transactions, how would you identify:

a. Identifying High-Value Customers

To identify high-value customers, I would:

- Use RFM analysis based on:
    - Recency: How recently they made a purchase.
    - Frequency: How often they purchase.
    - Monetary value: How much they spend.
- Rank customers on these metrics to segment them into tiers (for example: gold, silver, bronze).
- Optionally, we can use clustering (like K-Means) to group customers based on these behaviors.

b. Identifying Seasonality Trends

To detect seasonality:

- Aggregate transactions by week or month, and analyze the total spend or number of purchases over time.
- Plot this using a line chart to visualize peaks and drops.
- Use techniques like:
    - Moving averages to smooth the trend.
    - Decomposition (trend + seasonality + residual).
- Cross-check spikes with holidays, campaigns, or events that may drive purchases.

Section B: Python & Modelling

3. Python Task:

You're given a dataset with features: `user_id`, `last_login_days_ago`, `num_purchases`, `avg_purchase_value`, and a binary target `churned`.

a. Write Python code to prepare this data for a logistic regression model.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score

# sample simulated dataset
# in real case: df = pd.read_csv("your_dataset.csv")
df = pd.DataFrame({
    'user_id': range(1, 11),
    'last_login_days_ago': [5, 30, 2, 15, 60, 1, 45, 10, 3, 22],
    'num_purchases': [1, 0, 5, 2, 0, 8, 1, 4, 6, 3],
    'avg_purchase_value': [50, 0, 120, 70, 0, 150, 45, 100, 130, 85],
    'churned': [1, 1, 0, 0, 1, 0, 1, 0, 0, 0]
})

# step 1: define features and target
features = ['last_login_days_ago', 'num_purchases', 'avg_purchase_value']
target = 'churned'

X = df[features]
y = df[target]

# step 2: train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# step 3: standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# step 4: train logistic regression model
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
```

```
# step 5: evaluate model
y_pred = model.predict(X_test_scaled)
y_proba = model.predict_proba(X_test_scaled)[:, 1]

print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("ROC AUC Score:", roc_auc_score(y_test, y_proba))
```

Explanation:

Here is how I would prepare the data and evaluate the model for churn prediction:

1. Dataset Structure:
   I assumed the dataset contains user_id, behavioral and purchase-related features (last_login_days_ago, num_purchases, avg_purchase_value), and a binary column churned.

2. Feature Selection & Target:
   The input features help capture recency and value of user activity, while churned is our target variable (1 = churned, 0 = retained).

3. Preprocessing:
   I split the data into training and test sets and used StandardScaler to normalize the input features. This improves model performance and convergence.

4. Model:
   I used Logistic Regression a simple, interpretable model that performs well for binary classification problems like churn prediction.

5. Evaluation:
   The model's performance is evaluated using:
   - Classification Report: Shows precision, recall, and F1-score.
   - Confusion Matrix: Helps visualize true vs false positives/negatives.
   - ROC AUC Score: Measures the model's ability to distinguish churners from non-churners.

b. Briefly explain how you'd evaluate the model performance.

```
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("ROC AUC Score:", roc_auc_score(y_test, y_proba))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

Confusion Matrix:
 [[1 0]
 [0 1]]
ROC AUC Score: 1.0
```

To evaluate the logistic regression model, I used three key metrics:

1) Classification Report:
   This includes:
   - Precision: How many predicted churns were actually correct.
   - Recall: How many actual churns the model correctly identified.
   - F1-Score: The balance between precision and recall.

   In my case, all values were 1.00, indicating perfect predictions on the test set. However, this could be due to the small size of the test data.

2) Confusion Matrix:
   This shows how many predictions were true positives, false positives, true negatives, and false negatives.
   - The matrix output was:

     ```
     [[1 0]
      [0 1]]
     ```
   - This means the model correctly predicted both a churned and a non-churned user zero mistakes.

3) ROC AUC Score:
   This metric shows the model's ability to distinguish between churned and non-churned users.
   - A score of 1.0 indicates a perfect classifier, but again, this should be interpreted cautiously given the small sample size.

4. Machine Learning:

You are asked to build a customer segmentation model.

a. Which algorithm(s) would you use and why?

To build a customer segmentation model, I would use unsupervised learning, specifically the K-Means clustering algorithm.

- Simple and effective: K-Means is widely used for customer segmentation due to its simplicity and scalability.
- Well-suited for numeric customer features: It works well when we have continuous variables like spending behavior, frequency of purchases, or engagement levels.
- Easy to interpret: The resulting clusters can be used to label customers as, for example, “high value”, “occasional spenders”, or “at risk” based on common patterns.

If the data is complex or non-spherical, I might consider DBSCAN or Hierarchical Clustering, but I would start with K-Means.

b. What preprocessing would you do before modeling?

1. Feature Selection:
   I would select relevant behavioral and transactional features for example:
   - total_spend, transaction_count, avg_spend_per_transaction
   - login_frequency, redemption_rate

2. Handling Missing Values:
   Fill or drop any missing data to ensure clean input for clustering.

3. Feature Scaling:
   I would standardize all numeric features using StandardScaler or MinMaxScaler to ensure that features with large ranges (for example, spend) don’t dominate clustering.

4. Dimensionality Reduction:
   If there are many features, I’d apply PCA (Principal Component Analysis) to reduce dimensions while preserving important variance, which also helps visualize clusters in 2D.

5. Elbow Method or Silhouette Score:
   I would determine the optimal number of clusters by using the Elbow Method or Silhouette Score before fitting K-Means.

Section C: Applied Case

5. Business Case:
A product team wants to understand what factors lead to user conversion (first purchase).

a. What approach would you take to analyze this?
To understand what drives a user to make their first purchase, I would take the following analytical approach:

1. Define the Target
- Create a binary column:
  converted = 1 if the user has made a purchase, 0 otherwise.

2. Collect Features
Use available datasets to extract behavioral and demographic features before their first purchase:
- From engagement.csv: app opens, logins, redemptions
- From members.csv: age, tier, signup
- Optionally engineer time-based features: time between signup and first activity.

3. Exploratory Data Analysis (EDA)
Compare converted vs non-converted users:
- Do converters log in more often?
- Are certain tiers or signup channels more likely to convert?
- Use visualizations like bar charts and boxplots to explore patterns.

4. Modeling (optional)
- Train a simple Logistic Regression model to see which features are most predictive of conversion.
- Interpret model coefficients to understand feature importance.

b. How would you explain your findings to a non-technical stakeholder?

I would keep the explanation clear, and outcome driven:

> We looked at user behavior and demographics to understand what makes someone more likely to make their first purchase.
> We found that users who open the app frequently in the first week and who engage with redemptions early on are much more likely to convert.
> Additionally, users from higher-tier memberships or who signed up via referral channels also show higher conversion rates.
> This tells us that improving early engagement like encouraging app opens and offering incentives can significantly boost conversion.

I would support this with simple visuals, such as:

- A bar chart comparing conversion rates by signup channel
- A timeline showing the average time to first purchase