Go Global Travel – R&D Answers

Part 1:

1. Inside IncludeInFinalResults function, the switch(enforcedOption) uses enforcedOption the wrong way because enforcedOption is a variable. Instead of using: enforcedOption.All:, enforcedOption.AvailableOnly:, enforcedOption.NotAvailableOnly:, it should be refactored and use RoomAvailableOption.All instead of enforcedOption.All and the enum switch should look like this:

```
return enforcedOption switch
{
   RoomAvailableOption.All: return true,
   RoomAvailableOption.AvailableOnly: return  isAvailable,
   RoomAvailableOption.NotAvailableOnly: return  !isAvailable,
  default: return true;
};
```

2. GetUserFromDB refactoring:

```
public User GetUserFromDB(int userId)
{
    DataRow dr = DataRepository.GetUserById(userId);
    if (dr == null)
        return null;

    return new User
    {
        UserId = userId,
        FirstName = GetString(dr, "FirstName"),
        LastName = GetString(dr, "LastName"),
        Address = GetString(dr, "Address"),
        CityName = GetString(dr, "CityName"),
        CountryName = GetString(dr, "CountryName"),
        Email = GetString(dr, "Email")
    };
}

private string GetString(DataRow dr, string columnName)
{
    return dr.Table.Columns.Contains(columnName) &&
           dr[columnName] != DBNull.Value
           ? dr[columnName].ToString()
           : null;
}
```

Part 3:

1. The sql query that combines the tables AMOUNTS and ACCOUNTS
   And returns Amount and Currency:

```
SELECT
   acc.BANK_ID,
   acc.BRANCH_ID,
   acc.ACCOUNT_NUM,
   acc.CURRENCY,
   am.AMOUNT
FROM ACCOUNTS acc
JOIN (
   SELECT
      AMOUNT,
      PARSENAME(REPLACE(ACCOUNT_REC, ' ', '.'), 3) AS BANK_ID,
      PARSENAME(REPLACE(ACCOUNT_REC, ' ', '.'), 2) AS BRANCH_ID,
      PARSENAME(REPLACE(ACCOUNT_REC, ' ', '.'), 1) AS ACCOUNT_NUM
   FROM AMOUNTS
) am
ON acc.BANK_ID = am.BANK_ID
AND acc.BRANCH_ID = am.BRANCH_ID
AND acc.ACCOUNT_NUM = am.ACCOUNT_NUM;
```

2. The reasons why the query run time take longer than it should are:

   1. For every row in “TMP_TAB” table the query will run all over the rows of TMP_TAB and for each row will run over all the rows of MR_SMT or MR_OP, and that is a very big query.

   2. “BETWEEN date-7 AND date” is missing indexes for: (ID, BUSINESS_DATE) to make the query run faster.

   3. DISTINCT MAX() in “CALC_AMT” is not needed because MAX() is already selecting only 1 row. This makes the query take more time to finish running.

   4. After adding Indexes the query will run faster. Add indexes:

```
CREATE INDEX IDX_MR_SMT_ID_DATE ON MR_SMT(ID, BUSINESS_DATE);
CREATE INDEX IDX_MR_OP_ID_DATE  ON MR_OP(ID, BUSINESS_DATE);
CREATE INDEX IDX_TMP_TAB_ID_DATE ON TMP_TAB(ID, BUSINESS_DATE);
```

5. To optimize the query to run faster, the query can run without using the “CALC_AMT” function, then the query should run faster with optimization for parallel execution and join optimization. The new query should look like this:

```
SELECT DISTINCT
    T.BRANCH,
    T.ACCOUNT,
    CASE
        WHEN T.AMT_TYPE = 'C' THEN SMT.MAX_AMT
        ELSE OP.MAX_AMT
    END AS CALC_AMT,
    T.ID,
    T.FILENAME,
    T.BUSINESS_DATE,
    T.POPULATION_DATE
FROM TMP_TAB T
LEFT JOIN (
    SELECT
        ID,
        BUSINESS_DATE,
        MAX(AMT) AS MAX_AMT
    FROM MR_SMT
    GROUP BY ID, BUSINESS_DATE
) SMT
    ON SMT.ID = T.ID
   AND SMT.BUSINESS_DATE BETWEEN T.BUSINESS_DATE - 7 AND
T.BUSINESS_DATE
LEFT JOIN (
    SELECT
        ID,
        BUSINESS_DATE,
        MAX(AMT) AS MAX_AMT
    FROM MR_OP
    GROUP BY ID, BUSINESS_DATE
) OP
    ON OP.ID = T.ID
   AND OP.BUSINESS_DATE BETWEEN T.BUSINESS_DATE - 7 AND
T.BUSINESS_DATE;
```