HELPER FUNCTIONS (INSERT / UPDATE)

```
export async function updateDailyBioInfo(pool: any, user: number, date:
string, BMI?: number, HbA1c?: number, BloodGlucose?: number):
Promise<BiometricsInfo[] | null> {
      let queryString = `
      INSERT INTO diabetes.BIOMETRICSINFO (User_Id, BioEntryDate, BMI,
HbA1c, BloodGlucose)
      VALUES (${user}, '${date}', ${BMI}, ${HbA1c}, ${BloodGlucose})
      ON DUPLICATE KEY UPDATE
        BMI = VALUES(BMI),
        HbA1c = VALUES(HbA1c),
        BloodGlucose = VALUES(BloodGlucose)`;
    try {
        await pool.query(queryString);
    } catch (error) {
        console.error('Error creating new daily info: query execution
failure', error);
        console.error('Query string was:', queryString);
        return null;
    }
    let [confirmation] = await pool.query(`SELECT * FROM
diabetes.BIOMETRICSINFO WHERE User_Id = ${user} AND BioEntryDate =
'${date}'`);
    return confirmation as BiometricsInfo[];
}

export async function updateDailyCondInfo(pool: any, user: number, date:
string, Stroke?: number, HighChol?: number, HighBP?: number,
HeartDisease?: number, Hypertension?: number): Promise<ConditionsInfo[] |
null> {
    let queryString = `
      INSERT INTO diabetes.CONDITIONSINFO (User_Id, CondEntryDate, Stroke,
HighChol, HighBP, HeartDisease, Hypertension)
      VALUES (${user}, '${date}', ${Stroke}, ${HighChol}, ${HighBP},
${HeartDisease}, ${Hypertension})
      ON DUPLICATE KEY UPDATE
        Stroke = VALUES(Stroke),
```

```
        HighChol = VALUES(HighChol),
        HighBP = VALUES(HighBP),
        HeartDisease = VALUES(HeartDisease),
        Hypertension = VALUES(Hypertension)`;
    try {
        await pool.query(queryString);
    } catch (error) {
        console.error('Error creating new daily info: query execution
failure', error);
        console.error('Query string was:', queryString);
        return null;
    }
    let [confirmation] = await pool.query(`SELECT * FROM
diabetes.CONDITIONSINFO WHERE User_Id = ${user} AND CondEntryDate =
'${date}'`);
    return confirmation as ConditionsInfo[];
}

export async function updateDailyLifeInfo(pool: any, user: number, date:
string, Smoker?: number, CheckChol?: number, Fruits?: number, Veggies?:
number): Promise<LifestyleInfo[] | null> {
    let queryString = `
      INSERT INTO diabetes.LIFESTYLEINFO (User_Id, LifeEntryDate, Smoker,
CheckChol, Fruits, Veggies)
      VALUES (${user}, '${date}', ${Smoker}, ${CheckChol}, ${Fruits},
${Veggies})
      ON DUPLICATE KEY UPDATE
        Smoker = VALUES(Smoker),
        CheckChol = VALUES(CheckChol),
        Fruits = VALUES(Fruits),
        Veggies = VALUES(Veggies)`;
    try {
        await pool.query(queryString);
    } catch (error) {
        console.error('Error creating new daily info: query execution
failure', error);
        console.error('Query string was:', queryString);
        return null;
    }
```

```
    let [confirmation] = await pool.query(`SELECT * FROM
diabetes.LIFESTYLEINFO WHERE User_Id = ${user} AND LifeEntryDate =
'${date}'`);
    return confirmation as LifestyleInfo[];
}
```

TRANSACTION FUNCTIONS (ACID verified)

```
// Query to make use of first transaction
export async function updateBiometricsWithRiskAlerts(
  userId: number,
  date: string,
  bmi?: number,  // BIOMETRICSINFO attributes
  hba1c?: number,
  bloodGlucose?: number,
  heartDisease?: number, // CONDITIONSINFO attributes
  stroke?: number,
  highChol?: number,
  highBP?: number,
  hypertension?: number
): Promise<void> {
  const connection = await pool.getConnection();
  try {
    await connection.beginTransaction();

    const confirmBioUpdate = await updateDailyBioInfo(connection, userId,
date, bmi, hba1c, bloodGlucose);
    const confirmCondUpdate = await updateDailyCondInfo(connection,
userId, date, stroke, highChol, highBP, heartDisease, hypertension);
    if (!confirmBioUpdate || !confirmCondUpdate) {
      throw new Error('Failed to update biometrics or conditions info');
    }
    await connection.query(`
      INSERT INTO RiskAlerts (User_Id, Alert_Message, Created_On)
      SELECT DISTINCT u.User_Id,
        'Elevated risk detected: Review health conditions and consult your
physician.',
        NOW()
      FROM USERINFO u
      JOIN BIOMETRICSINFO b ON u.User_Id = b.User_Id
      JOIN CONDITIONSINFO c ON u.User_Id = c.User_Id
```

```
      WHERE u.User_Id = ?
      AND (b.BMI > 30
        OR b.HbA1c > 6.4
        OR b.BloodGlucose > 125
        OR c.HeartDisease = 1
        OR c.HighBP = 1)`,
      [userId]
    );

    await connection.query(
      'INSERT INTO UpdateLog (User_Id, Updated_On, Update_Reason) VALUES
(?, NOW(), ?)',
      [userId, 'Daily biometrics update with risk evaluation']
    );

    await connection.commit();
  } catch (error) {
    console.log(349, "Biometrics and Conditions transaction failed");
    await connection.rollback();
    throw error;
  } finally {
    connection.release();
  }
}
// Query to make use of second transaction
export async function updateLifestyleWithRewards(
  userId: number,
  date: string,
  veggies?: number,
  fruits?: number,
  smoker?: number,
  checkChol?: number
): Promise<void> {
  const connection = await pool.getConnection();
  try {
    await connection.beginTransaction();

    const confirmLifeUpdate = await updateDailyLifeInfo(connection,
userId, date, smoker, checkChol, fruits, veggies);
    if (!confirmLifeUpdate) {
```

```
      throw new Error('Failed to update lifestyle info');
    }

    await connection.query(`
      INSERT INTO Incentives (User_Id, Incentive_Description, Granted_On)
      SELECT DISTINCT l1.User_Id, 'Healthy Lifestyle Reward', NOW()
      FROM LIFESTYLEINFO l1
      JOIN LIFESTYLEINFO l2 ON l1.User_Id = l2.User_Id
      WHERE l1.User_Id = ?
      AND l1.LifeEntryDate = ?
      AND l2.LifeEntryDate < l1.LifeEntryDate
      AND (l1.Veggies > l2.Veggies
        OR l1.Fruits > l2.Fruits
        OR l1.CheckChol > l2.CheckChol
        OR l1.Smoker < l2.Smoker)
      ORDER BY l2.LifeEntryDate DESC
      LIMIT 1`,
      [userId, date]
    );

    await connection.query(
      'INSERT INTO UpdateLog (User_Id, Updated_On, Update_Reason) VALUES
(?, NOW(), ?)',
      [userId, 'Lifestyle update and improvement evaluation']
    );

    await connection.commit();
  } catch (error) {
    console.log(402, "Lifestyle and Incentives transaction failed");
    await connection.rollback();
    throw error;
  } finally {
    connection.release();
  }
}
```