Scenario:
Money Bank's Postgres database contains the following two tables (and their fields):

```
Customer
    * customer_id (number)
    * first_name (varchar(20))
    * last_name (varchar(20))
    * date_of_birth (timestamptz)
```

and

```
Transactions
    * txn_id (number)
    * customer_id (foreign-key indexed to Customer table)
    * txn_type (varchar(10))
    * txn_amount (number)
    * transaction_date (timestamptz)
```

Note that in the definition above,
- `transaction_type` is of type *CREDIT* or *DEBIT*
- Dates in the database are stored in *YYYY-MM-DD* format.

Using `python 3.8`**, and the postgresql package** `psycopg2`
(**NOTE**: **DO NOT** use data-science packages like `numpy`, `pandas`, etc. or frameworks like `django`, `flask`, `SqlAlchemy` etc. Simple python functions are sufficient for this project.)

Write a function **`calculate_savings(events,context)`** and any other associated helper functions to do the following:
1. Reads the `date` from the incoming `events` payload.
2. Reads all the transactions from the date provided above and for all customers that have transacted on that date and calculates the savings (credit - debit) for each customer.
3. Using the savings data gathered from step 2, the function returns a json payload in the following format:

```
{
  "statusCode": 200,
  "data": {
            25: 10,
            28: 100,
            ..
          }
}
```

In case of error, it returns the following payload:

```
{
  "statusCode": 400,
  "message": error-message
}
```

4. The `data` field above contains the *key-value* pairs in the format: `{ age: avg_saving }`

5. "`age`" is the age of each customer returned in step 2. Age is calculated from the customer's `date_of_birth` field and rounded to the nearest integer. Eg. In the row `25: 10` above, `25` contains the data for all customers whose rounded age is 25.
6. `avg_saving` is calculated as the average savings of all the customers in that age group and rounded to the nearest integer. So, in the example '`25: 10`' means that the average saving is 10 for customers whose rounded age is 25.
7. As indicated, you should have code to catch exceptions or errors that may happen.

Usage
The function is called with an events dictionary object:

```
events = {
     "database": "db_name",
     "port": db_port,
     "host": "db_endpoint",
     "username": "db_username",
     "password": "db_password",
     "date": "dd/mm/yyyy"
   }
```

The first 5 fields are used to connect to the db using `psycopg2`.

NOTE:
1. You should have a .py file containing all the functions
2. You should have a requirements.txt file containing all the packages you use.