```c
# include <stdio.h>
# include <stdlib.h>
# include <string.h>

#define MAX 50


typedef struct Data {
    char num[MAX];
    char name[MAX];
    char s_name[MAX];
    unsigned short int age;
    char job[MAX];
    float pay;
} DB;


int upload(char [], DB []);
void printDB(DB [], int);
void ageSelect(DB [], char [], int);
void IDSelect(DB [], char [], char [], int);
void jobSelect(DB [], char [], char [], int);



/*****************************************************************************************/
/* A menu to choose the operation to perform on the dataset. */

int main (int argc, char *argv[]) {
    if (argc != 3) {
        printf("\nError: Invalid Argument Number!\n");
        return 0;
    }

    printf("\nQueries:\n");
    printf("\t1 - People older than n years of age.\n");
    printf("\t2 - Personal data corresponding to an ID number.\n");
    printf("\t3 - People with the same occupation.\n");

    unsigned short int choice;
    printf("\nQuery: ");
    scanf("%d", &choice);

    DB dbData[MAX];
    int dbSize = upload( *(argv + 1) , dbData);
    char w[MAX];

    if (choice == 1) {
        getchar();
        ageSelect(dbData , *(argv + 2) , dbSize);
    }
    else if (choice == 2) {
        printf ("ID number: ");
        getchar();
        gets(w);
        IDSelect( dbData , *(argv + 2) , w , dbSize);
    }

```

```c

    else if (choice == 3) {
        printf("Occupation: ");
        getchar();
        gets(w);
        jobSelect(dbData, *(argv + 2), w, dbSize);
    }
    else {
        printf ("Incorrect Query!\n");
        return 0;
    }
    printf ("\nResults in %s", *(argv + 2));
}


/*****************************************************************************/

/* Reads a dataset from a text file. */

int upload (char s[], DB dbData []) {
    FILE *fin;
    int i = 0;

    fin = fopen (s, "r");

    while ( fscanf(fin, "%s %s %s %d %s %f\n", &dbData[i].num    , &dbData[i].name ,
                                               &dbData[i].s_name , &dbData[i].age  ,
                                               &dbData[i].job    , &dbData[i].pay) == 6
                                               && i < MAX)
        i++;
    return i;
}


/*****************************************************************************/

/* Prints the dataset on a text file - if needed for some reason...*/

void printDB(DB sorted[], int i) {
    FILE * fout = fopen("dbRead.txt", "w");

    for (int j = 0; j < i; j++)
        fprintf(fout, "%4s  %-20s %-20s %3d   %-15s %8.2f\n",
            sorted[j].num, sorted[j].name,
            sorted[j].s_name, sorted[j].age,
            sorted[j].job, sorted[j].pay);
}
```

```c
/*******************************************************************************/

/* Extract all the entries basing on the age given as input at runtime. */

void ageSelect(DB dbData[], char fileOut[], int dbSize) {
    unsigned short int age;
    printf("\nAge: ");
    scanf("%d" , &age);

    DB sorted[MAX];
    FILE * fout;
    fout = fopen(fileOut , "w");
    int i = 0;

    for (int j = 0 ; j < dbSize ; j++) {
        if (dbData[j].age >= age)
            sorted[i++] = dbData[j];
    }

    if (i == 0) {
        printf("No match for age >= %d\n", age);
        fprintf(fout, "No match for age ≥ %d", age);
    }
    else if (i == 1) {
        fprintf(fout , "A single match for age ≥ %d:\n\n", age);
        fprintf(fout , "%4s  %-20s %-20s %3d   %-15s %8.2f\n",
                       sorted[0].num    , sorted[0].name,
                       sorted[0].s_name , sorted[0].age,
                       sorted[0].job    , sorted[0].pay );
    }
    else {
        fprintf(fout, "%d matches for age ≥ %d:\n\n", i, age);
        for (int j = 0 ; j < i ; j++)
            fprintf(fout , "%4s  %-20s %-20s %3d   %-15s %8.2f\n",
                           sorted[j].num    , sorted[j].name,
                           sorted[j].s_name , sorted[j].age,
                           sorted[j].job    , sorted[j].pay );
    }

}
```

```c
/*****************************************************************************/

/* Search for a single entry for a given ID. */

/* IMPORTANT: Assumes that the elements are sorted by ID number because it performs a
              binary search in the dataset. */

void IDSelect(DB dbData [], char fileOut[], char ID[], int dbSize) {
    FILE * fout;
    fout = fopen(fileOut, "w");

    int cond, low = 0, high, mid;
    high = dbSize - 1;

    while (low <= high) {
        mid = (low + high) / 2;
        if ( (cond = strcmp (ID, dbData[mid].num)) < 0 )
            high = mid - 1;
        else if (cond > 0)
            low = mid + 1;
        else {  /*found*/
            fprintf(fout, "Data for ID = %s:\n\n", ID);
            fprintf(fout, "%-20s %-20s %3d   %-15s %8.2f",
            dbData[mid].name , dbData[mid].s_name , dbData[mid].age,
            dbData[mid].job  , dbData[mid].pay);
            return;
        }
    }
    printf("No match found.\n");
    fprintf(fout, "\nNo match found.");
}




/*****************************************************************************/

/* Select all entries with the same job. */


void jobSelect (DB dbData [], char fileOut[], char job[], int dbSize) {
    FILE * fout;
    fout = fopen (fileOut , "w");
    DB positions[MAX];

    /*Linear search*/
    int k = 0;
    for (int i = 0 ; i < dbSize ; i++) {
        if ((strcmp (job, dbData[i].job)) == 0)
            positions[k++] = dbData[i];
    }

    if (k == 0) {
        printf("\nNo match found for \"%s\"\n", job);
        fprintf(fout, "\nNo match found for \"%s\"\n", job);
    }


```

```c
    else if (k == 1)
        fprintf (fout, "\nOne match found for \"%s\"\n\n", job);
    else
        fprintf (fout, "%d matches found for \"%s\"\n\n", k , job);

    for (int j = 0 ; j < k ; j++)
        fprintf (fout, "%4s  %-20s %-20s %3d  %8.2f\n",
                        positions[j].num    , positions[j].name ,
                        positions[j].s_name , positions[j].age  ,
                        positions[j].pay );
}
```