Project

Part B

הקדמה

בחלק זה של הפרויקט עליכם להקים אתר למערכת המבוססת על החלק הראשון של הפרויקט, בהתאם להוראות המפורטות בעמודים הבאים.

עליכם לבנות את המערכת באמצעות אפליקציה יחידה ששמה Stocks_App.

אין להשתמש בשם אפליקציה אחר! שימו לב לכך ש-S ו-A נכתבות כ-capital letters.

החיבור למסד הנתונים יתבצע בהתאם להנחיות המופיעות בתרגול 8. שימו לב כי חלק משמות הרלציות/שדות עשוי להפוך ל-lowercase.

לאחר היווצרות ה-models באמצעות הפקודה inspectdb יש לשנות את השורה הבאה (ברלציה buying):

```
symbol = models.ForeignKey('Stock', models.DO_NOTHING, db_column='Symbol',
        to_field='tDate', related_name='buying_symbol_set')
```

לשורה הבאה (כלומר להוסיף את הארגומנט עם related_name ולהציב בו את הערך '+'):

```
symbol = models.ForeignKey('Stock', models.DO_NOTHING, db_column='Symbol',
        related_name='+')
```

תיאור המערכת

הרלציות בהן תשתמשו בחלק זה של הפרויקט מוגדרות בהשראת הסיפור מחלק א', אך ייתכנו מספר הבדלים. לפני תחילת העבודה על התרגיל, מחקו את כל הרלציות שיצרתם עבור חלק א' של הפרויקט. במודל מופיעים קבצי csv (בקובץ הזיפ ששמו ProjectB_Files). מומלץ לעבור על קבצים אלו לפני תחילת העבודה. לקובץ הזיפ מצורף קובץ בשם create_commands.sql המכיל פקודות ליצירת הטבלאות במסד. עליכם להריצו דרך ה-console לפני תחילת העבודה על האתר. יש להקפיד על סדר ההרצה כפי שמופיע בקובץ.

הרלציות בהן תשתמשו בחלק זה הן:

Investor (ID, Name, Amount)

Company (Symbol, Sector, Location, Founded)

Stock (Symbol, tDate ,Price)

Buying (tDate , ID, Symbol, BQuantity)

Transactions (tDate, ID, TAmount)

עבור כל אחת מהרלציות, ה-primary key מסומן באמצעות קו תחתון.

תיאור הרלציות:

Investor– רלציה הכוללת מידע אודות משקיעים.

- ID – מספר הזהות של המשקיע.
- Name- שמו של המשקיע.
- Amount - סכום הכסף בחשבון ההשקעות של המשקיע הפנוי לצורך קניית מניות חדשות. מדובר בערך שעשוי להשתנות ובאחריותכם לתחזק אותו בהתאם לדרישות המופיעות בהמשך התרגיל.

Company– רלציה הכוללת מידע אודות חברות.

- Symbol – מזהה החברה.
- Sector – המגזר אליו משתייכת החברה.
- Founded- השנה בה הוקמה החברה.
- Location- המדינה בה יושב המטה הראשי של החברה.

Stock– רלציה הכוללת מידע אודות מניות של חברות.

- tDate – תאריך.
- Symbol – מזהה החברה.
- Price- ערך המניה של החברה.

הערה: בדומה לחלק א' של הפרויקט, גם כאן יש להניח כי למניה יש ערך יחיד ביום כלשהו (כלומר בשונה מהמציאות המחיר לא משתנה לאורך היום).

Buying– רלציה הכוללת תיעודי קניית מניות על ידי משקיע.

- tDate – תאריך.
- ID - מספר הזהות של המשקיע.
- Symbol – מזהה החברה.
- BQuantity – כמות המניות של החברה שרכש המשקיע בקניה זו.

הערה: שימו לב שמשקיע יכול לבצע רכישה של מניות אך ורק אם סכום הכסף הפנוי שברשותו (AvailableCash) גבוה לפחות כמו ערך הקניה שברצונו לבצע (ערך המניה של החברה באותו היום כפול כמות המניות שברצונו לרכוש). יש לוודא דרישה זו בהתאם לדרישות המתוארות בהמשך התרגיל.
הניחו לצורך פשטות כי לא ניתן למכור מניות שנקנו.
הערה: אל חשש. בחלק א' של הפרויקט לא נדרשתם להתייחס לדרישה זו.

Transactions– רלציה הכוללת תיעודי העברת כסף מחשבון הבנק של המשקיע לחשבון ההשקעות שלו.

- tDate – תאריך.
- ID - מספר הזהות של המשקיע.
- TAmount – כמות הכסף שהעביר המשקיע לחשבון ההשקעות שלו.

להלן תיאור הדפים השונים אשר אמור להכיל האתר שתבנו :

1. דף הבית

דף הבית של האתר צריך לכלול את הפרטים הבאים :

- כותרת לאתר המערכת.
- תמונה מייצגת שמתאימה לאתר (לבחירתכם)
- קישורים לשאר העמודים אשר מתוארים בגיליון. מיקומם של הקישורים בעמוד לא קריטי כל עוד הם בולטים לעין עבור המשתמש. **יש ליצור בכל עמוד (לא רק בעמוד הבית) קישור לכל אחד מהעמודים האחרים**.

להלן דוגמה לעמוד :

TechniStoncks - Israel Website of Stocks

Query Results

Add Transaction

Buy Stocks

2. הצגת תוצאות שאילתות

לפני תחילת העבודה על עמוד זה מומלץ לקרוא את הנספח המופיע בסוף הנחיות התרגיל.

בעמוד זה יש ליצור, בנוסף לקישורים המובילים לעמודים האחרים, טבלה עבור כל אחת משלוש השאילתות הבאות, אשר בה יוצגו תוצאות השאילתה. יש להוסיף כותרת לכל אחת מטבלאות המודיעה על מספר השאילתה (ראו דוגמה בעמוד הבא).

הערה: בחלק זה אתם יכולים להשתמש בשאילתות VIEW. אם ברצונכם לעשות זאת, עליכם לכתוב אותן בקובץ SQL בשם view_queries ולהגיש אותן יחד עם שאר קבצי התרגיל כפי שמפורט בהנחיות ההגשה שבסוף מסמך זה.

עבור כל שאילתה מותר להשתמש ב-4 שאילתות VIEW לכל היותר.

a. "משקיע מגוון" מוגדר כמשקיע אשר קיים יום כלשהו בו הוא רכש מניות של חברות מלפחות 6 סקטורים שונים.

החזירו את שמות כל המשקיעים המגוונים ואת הסכום הכולל שהוציאו על קניית מניות בתקופה המתועדת במסד (עם דיוק של 3 ספרות אחרי הנקודה). יש להחזיר את המידע בצורה ממוינת בסדר יורד לפי הסכום.

דוגמה: נתונה טבלת הקניות הבאה:

tDate	ID	Symbol	BQuantity
2021-12-16	1234	BLL	5
2021-12-17	1234	BLL	4
2021-12-17	1234	ADBE	3

ונניח כי מחיר המניה BLL ב-16 וב-17 בדצמבר הוא 120 ו-130 דולר למניה בהתאמה, וכי מחיר מניית ADBE ב-17 בדצמבר הוא 100 דולר. במקרה זה הסכום שהוציא משקיע 1234 על קניית מניות בתקופה המתועדת בטבלה הוא:

$$5\cdot 120+4\cdot 130+3\cdot 100=1420$$

הערה: ניתן להשתמש באופרטור ROUND כדי להורות על החזרת שלוש ספרות אחרי הנקודה.

b. "חברה פופולרית" מוגדרת כחברה אשר מקיימת את שני התנאים הבאים:

i. **בכל** אחד מימי המסחר המתועדים במסד (השדה tDate ברלציה Buying) נרכשה לפחות מניה אחת שלה.

ii. **אף** חברה נוספת שמשתייכת לסקטור שלה לא מקיימת את תנאי i.

עבור כל חברה פופולרית יש להחזיר את המזהה שלה, את **שמו** של המשקיע אשר מחזיק ב**כמות** המניות הגדולה ביותר של אותה החברה ואת כמות זו.

יש להחזיר את המידע בצורה ממוינת בסדר עולה לפי מזהה החברה. במקרה של חברה שקיימים עבורה מספר משקיעים שעונים על התנאי (כלומר משקיעים שמחזיקים באותה כמות מניות וזוהי הכמות הגדולה ביותר של מניות החברה שמשקיע כלשהו מחזיק) יש להחזיר את השמות של כולם (כאשר השדה של שם מהווה קטגורית מיון משנית, בסדר עולה).

c. "חברה רווחית" היא חברה שמחיר המניה שלה ביום האחרון המתועד במסד (השדה tDate ברלציה Stock) גבוה ביותר מ-6% ממחיר המניה שלה ביום הראשון המתועד במסד.
החזירו עבור כל חברה רווחית את מספר המשקיעים שביצעו קניית מניות שלה ביום המסחר הראשון המתועד במסד (השדה tDate ברלציה Stock). יש להחזיר את התוצאה ממוינת בסדר עולה לפי מזהה החברה.

מומלץ לטעון למסד הנתונים שלכם את קבצי ה-CSV המצורפים למסד כדי לבדוק את נכונות הקוד שלכם.

להלן דוגמה לעמוד (אלו הן התשובות שאתם אמורים לקבל עבור הקבצים הנתונים, אם כי בבדיקת התרגיל נשתמש בקבצים המכילים נתונים אחרים):

TechniStoncks - Israel Website of Stocks

Qurery 1:

Name	Total Sum
Roger Hood	273042.369
Patricia Crumedy	237127.997
Patricia Williams	166696.56
Cynthia Owens	135983.652
Jill Jones	132209.75
Larry Saran	107015.94

Qurery 2:

Symbol	Name	Quantity
COST	David Dunn	21
COST	Thomas Escobar	21

Qurery 3:

Symbol	Buyers Number
AVGO	2
AZO	2
DPZ	1
MLM	4
NVDA	1
URI	0

Home Page

Add Transaction

Buy Stocks

3. טופס ביצוע טרנזקציה

בעמוד זה יוכל המשתמש להכניס תיעוד חדש של העברת כסף מחשבון הבנק של משקיע המתועד במסד לחשבון ההשקעות שלו. לאחר ביצוע הטרנזקציה יתעדכן סכום הכסף הפנוי בחשבון ההשקעות של הלקוח (השדה Amount). יש לתעד את העברת הכסף ברלציה Transactions כך שתאריך ההעברה יוגדר כתאריך של היום האחרון המתועד ברלציה Stock. שימו לב כי בדאטה אותו קיבלתם ובדאטה עליו תיבחנו לא מתועדות בהתחלה קניית מניות המתרחשת בתאריך זה.

המשתמש יזין באמצעות טופס את מספר הזהות של המשקיע ואת סכום הכסף המבוקש. לאחר לחיצה על כפתור הגשת הטופס יוחזר המשתמש לעמוד זה. עליכם לאכוף ברמת ה-HTML את חובת מילוי שני שדות אלו.

כמו כן, בעמוד תופיע טבלה שתציג את 10 ההעברות האחרונות שמתועדות ברלציה Transactions (שדה מיון משני הוא מספר הזהות של מבצע ההעברה, בסדר יורד).

לאחר הגשת הטופס ולפני עדכון הרלציה Transactions יש לוודא כי מספר הזהות של המשקיע אכן קיים במסד. אם לא, יש להציג הודעת שגיאה המודיעה על כך ולא לבצע עדכון של הרלציה.

ניתן לבצע רק העברה אחת לחשבונו של משקיע ביום כלשהו.
על כן, במקרה שמנסים להזין טרנזקציה נוספת עבור משקיע בתאריך המוגדר לעיל (התאריך של היום האחרון המתועד ברלציה Stocks) הפעולה לא תאושר ותופיע הודעת שגיאה שמכריזה על הסיבה לכך.

להלן דוגמה לעמוד:

TechniStoncks - Israel Website of Stocks

Add New Transaction

ID:
Transaction Sum:
Submit

Last 10 Transactions:

Date	Investor ID	Transaction Sum
Feb. 29, 2024	782224519	5
Feb. 29, 2024	641939476	13
Feb. 29, 2024	411150634	0
Feb. 28, 2024	782224519	4482
Feb. 28, 2024	641939476	3284
Feb. 28, 2024	445746760	4377
Feb. 28, 2024	411150634	1013
Feb. 28, 2024	389136634	1666
Feb. 28, 2024	209765575	3780
Feb. 27, 2024	534280104	4358

Home Page

Query Results

Buy Stocks

4. טופס ביצוע קניית מניות

בעמוד זה יוכל המשתמש להכניס תיעוד חדש של קניית מניות של חברה כלשהי על ידי משקיע. לאחר שמתבצעת קניה יש לחסר את עלותה (כמות המניות כפול ערך המניה) מערך סכום הכסף הפנוי בחשבונו של המשקיע. יש לתעד את ביצוע הקניה ברלציה Buying כך שתאריך ההעברה יוגדר כתאריך של היום האחרון המתועד ברלציה Stock. שימו לב כי בדאטה אותו קיבלתם ובדאטה עליו תיבחנו לא מתועדות בהתחלה קניית מניות המתרחשת בתאריך זה.

המשתמש יזין באמצעות טופס את מספר הזהות של המשקיע, את מזהה החברה המבוקשת ואת כמות המניות של אותה החברה שברצונו לרכוש. לאחר לחיצה על כפתור הגשת הטופס יוחזר המשתמש לעמוד זה.

לאחר הגשת הטופס ולפני עדכון הרלציה Buying יש לוודא כי מספר הזהות של המשקיע וכי מזהה החברה אכן קיימים במסד. כמו כן, יש לוודא כי עלות הקניה (ערך המניה של החברה באותו היום כפול כמות המניות שנרכשו) **לא** גדולה יותר מסכום הכסף הפנוי שברשותו של המשקיע (השדה Amount ברלציה Investor). אם אחד מהתנאים הללו לא מתקיים יש להדפיס הודעת שגיאה המציינת מהם התנאים שהופרו (ייתכן כמה תנאים הופרו ביחד) ולא לבצע עדכון של הרלציה.

משקיע לא יכול לבצע פעולות קניות מרובות של מניות עבור אותה החברה ביום כלשהו.
יש להדפיס הודעת שגיאה המתריאה על כך במקרה שהמשתמש מנסה להזין רשומה שסותרת את תנאי זה.

בנוסף לטופס, בעמוד תופיע טבלה שתציג את 10 פעולות הקניה האחרונות המתועדות ברלציה Buying (שדה מיון משני הוא מספר הזהות של מבצע ההעברה בסדר יורד, ושדה מיון משני נוסף הוא מזהה החברה בסדר עולה).

להלן דוגמה לעמוד:

TechniStoncks - Israel Website of Stocks

Buy Stocks

ID:
Company:
Quantity:
Submit

Last 10 Stock Buys:

Date	Investor ID	Symbol	Quantity
Feb. 29, 2024	957317285	Apple	2
Feb. 29, 2024	957317285	COST	1
Feb. 29, 2024	411150634	COST	2
Feb. 28, 2024	997395948	COST	9
Feb. 28, 2024	997395948	ROP	6
Feb. 28, 2024	957317285	AZO	12
Feb. 28, 2024	957317285	ELV	6
Feb. 28, 2024	957317285	INTU	2
Feb. 28, 2024	957317285	IT	3
Feb. 28, 2024	957317285	MPWR	11

Home Page

Query Results

Add Transaction

הגדרת הפרויקט:

settings.py

```python
"""
Django settings for Project_PartB project.

Generated by 'django-admin startproject' using Django 5.0.2.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.0/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-z#5&5n-^wvjora(m(9gvgt$!g@sijr--at00(7c@t%c5(&%03)'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'Stocks_App',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'Project_PartB.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates']
        ,
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

הקבצים בפרויקט:

- Project_PartB C:\Users\liorb\
  - Project_PartB
    - __init__.py
    - asgi.py
    - settings.py
    - urls.py
    - wsgi.py
  - static
    - image.jpg
    - style.css
  - Stocks_App
    - migrations
    - __init__.py
    - admin.py
    - apps.py
    - models.py
    - tests.py
    - urls.py
    - views.py
  - templates
    - Add_Transaction.html
    - Buy_Stocks.html
    - index.html
    - Query_Results.html
  - manage.py
- External Libraries
- Scratches and Consoles

המשך – הגדרת הפרויקט:

```
WSGI_APPLICATION = 'Project_PartB.wsgi.application'


# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'mssql',
        'NAME': 'ben0lior',
        'USER': 'ben0lior',
        'PASSWORD': 'Qwerty12!',
        'HOST': 'techniondbcourse01.database.windows.net',
        'PORT': '1433',
        'OPTIONS': {"driver": "ODBC Driver 17 for SQL Server",}
    },
}


# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


```

```
# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = 'static/'
STATICFILES_DIRS = [
    BASE_DIR/'static'
]

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

urls(של הפרויקט):

urls.py

```python
"""
URL configuration for Project_PartB project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
import ...

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('Stocks_App.urls'))
]
```

תיקיית static:

style.css

```css
body{
    background-color: lightskyblue;
}

h1{
    color: brown;
    text-align: center;
}

h2{
    color: brown;
    text-align: center;
}

a{
    display: block;
    text-align: center;
    color: black;
    font-size: 25px;
}

.image-container {
    text-align: center;
}


table{
    margin: 0 auto;
    text-align: center;
}

form{
    text-align: center;
    margin: 0 auto;
}
```

תיקיית templates:

עמוד – index – HTML:

index.html

```
s\Project_PartB  load static %}
<head>
    <meta charset="UTF-8">
    <title>Home Page</title>
</head>
<link rel="stylesheet" href="{% static 'style.css' %}">
<h1>TechniStoncks - Israel Website of Stocks</h1>

<a href="Query_Results">Query Results</a><br>
<a href="Add_Transaction">Add Transaction</a><br>
<a href="Buy_Stocks">Buy Stocks</a><br><br>
<div class="image-container">
    <img src="{% static "image.jpg" %}" alt="Buy_or_Sell" style="..."/>
</div>
```

עמוד Quary_Results – HTML:

Query_Results.html

```
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Query Result</title>
</head>
<body>
<link rel="stylesheet" href="{% static 'style.css' %}">
<h1>TechniStoncks - Israel Website of Stocks</h1>
<h2>Qurery 1:</h2>
<table border="=1" width="40%">
    <th>Name</th>
    <th>Total Sum</th>
    {% for Q1 in sql_res %}
        <tr>
            <td>{{ Q1.Name }} <br></td>
            <td>{{ Q1.TotalSum }}<br></td>
        </tr>
    {% endfor %}
</table><br><br>

<h2>Qurery 2:</h2>
<table border="=1" width="40%">
    <th>Symbol</th>
    <th>Name</th>
    <th>Quantity</th>
    {% for Q2 in sql_res2 %}
        <tr>
            <td>{{ Q2.Symbol }} <br></td>
            <td>{{ Q2.Name }}<br> </td>
            <td>{{ Q2.Quantity }} <br></td>
        </tr>
    {% endfor %}
</table><br><br>

```

המשך - עמוד Quary_Results – HTML:

```
<h2>Qurery 3:</h2>
<table border="=1" width="40%">
    <th>Symbol</th>
    <th>Buyers Number</th>
    {% for Q3 in sql_res3 %}
        <tr>
            <td>{{ Q3.Symbol }}<br> </td>
            <td>{{ Q3.TotalBuyers }} <br></td>
        </tr>
    {% endfor %}
</table><br><br>

<a href="index">Home Page</a><br>
<a href="Add_Transaction">Add Transaction</a><br>
<a href="Buy_Stocks">Buy Stocks</a><br><br>

</body>
</html>
```

עמוד HTML – Add_Transaction:

Add_Transaction.html

```html
{% load static %}
<head>
    <meta charset="UTF-8">
    <title>Add Transaction</title>
</head>
<link rel="stylesheet" href="{% static 'style.css' %}">
<h1>TechniStoncks - Israel Website of Stocks</h1>
<h2>Add New Transaction</h2>
<form method="POST">
    {% csrf_token %}
    ID: <textarea name="ID" rows="1" cols="50" required></textarea><br>
    Transaction Sum: <textarea name="Transaction" rows="1" cols="50" required></textarea><br>
    <input type="submit">
</form>
<br>
{% if first_try == 1 %}
    {% if is_ID == 1 %}
        {% if is_not_Tran == 0 %}
            <h2>You have already made a transaction today</h2>
        {% endif %}
    {% else %}
        <h2>Investor ID does not exist</h2>
    {% endif %}
{% endif %}
<br>

<h2>Last 10 Transactions:</h2>
<table border="=1" width="40%">
    <th>Date</th>
    <th>Investor ID</th>
    <th>Transaction Sum</th>
    {% for Q4 in sql_res4 %}
        <tr>
            <td>{{ Q4.tDate }} <br></td>
            <td>{{ Q4.ID }}<br></td>
            <td>{{ Q4.TAmount }}<br></td>
        </tr>
    {% endfor %}
</table><br><br>

<a href="index">Home Page</a><br>
<a href="Query_Results">Query Results</a><br>
<a href="Buy_Stocks">Buy Stocks</a><br><br>
```

עמוד HTML – Buy_Stocks:

Buy_Stocks.html

```html
{% load static %}
<head>
    <meta charset="UTF-8">
    <title>Buy Stocks</title>
</head>
<link rel="stylesheet" href="{% static 'style.css' %}">
<h1>TechniStoncks - Israel Website of Stocks</h1>
<h2>Buy Stocks</h2>
<form method="POST">
    {% csrf_token %}
    ID: <textarea name="ID" rows="1" cols="50"></textarea><br>
    Company: <textarea name="Symbol" rows="1" cols="50"></textarea><br>
    Quantity: <textarea name="Quantity" rows="1" cols="50"></textarea><br>
    <input type="submit">
</form>
<br>
{% if first_try == 1 %}
    {% if is_ID == 0 %}
        {% if is_symbol == 0 %}
            <h2>ID and Symbol does not exist</h2>
        {% else %}
            <h2>ID does not exist</h2>
        {% endif %}
    {% elif is_symbol == 0 %}
        <h2>Symbol does not exist</h2>
    {% endif %}
    {% if is_ID == 1 and is_symbol == 1 %}
        {% if is_large == 0 %}
            {% if is_not_Buy == 0 %}
                <h2>You have already bought this stock today</h2>
                <h2>and you don't have enough money in your trading account</h2>
            {% else %}
                <h2>you don't have enough money in your trading account</h2>
            {% endif %}
        {% elif is_not_Buy == 0 %}
            <h2>You have already bought this stock today</h2>
        {% endif %}
    {% endif %}
{% endif %}


<br>
<h2>Last 10 Stock Buys:</h2>
<table border="=1" width="40%">
    <th>Date</th>
    <th>Investor ID</th>
    <th>Symbol</th>
    <th>Quantity</th>
    {% for Q5 in sql_res5 %}
        <tr>
            <td>{{ Q5.tDate }} <br></td>
            <td>{{ Q5.ID }}<br></td>
            <td>{{ Q5.Symbol }}<br></td>
            <td>{{ Q5.BQuantity }}<br></td>
        </tr>
    {% endfor %}
</table><br><br>

<a href="index">Home Page</a><br>
<a href="Query_Results">Query Results</a><br>
<a href="Add_Transaction">Add Transaction</a><br><br>
```

תיקיית האפליקציה(Stocks_App):

apps:

```python
from django.apps import AppConfig


class StocksAppConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'Stocks_App'
```

urls(של האפליקציה):

```python
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('index', views.index, name='index'),
    path('Query_Results', views.Query_Results, name='Query_Results'),
    path('Add_Transaction', views.Add_Transaction, name='Add_Transaction'),
    path('Buy_Stocks', views.Buy_Stocks, name='Buy_Stocks'),
]
```

models:

```python
# This is an auto-generated Django model module.
# You'll have to do the following manually to clean this up:
#   * Rearrange models' order
#   * Make sure each model has one field with primary_key=True
#   * Make sure each ForeignKey and OneToOneField has `on_delete` set to the desired behavior
#   * Remove `managed = False` lines if you wish to allow Django to create, modify, and delete the table
# Feel free to rename the models, but don't rename db_table values or field names.
from django.db import models


class Buying(models.Model):
    # Field name made lowercase. The composite primary key (tDate, ID, Symbol) found, that is not supported. The first column is selected.
    tdate = models.OneToOneField(to='Stock', models.DO_NOTHING, db_column='tDate', primary_key=True)
    id = models.ForeignKey(to='Investor', models.DO_NOTHING, db_column='ID')  # Field name made lowercase.
    symbol = models.ForeignKey(to='Stock', models.DO_NOTHING, db_column='Symbol', related_name='+')
    bquantity = models.IntegerField(db_column='BQuantity', blank=True, null=True)  # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'Buying'
        unique_together = (('tdate', 'id', 'symbol'),)


class Company(models.Model):
    symbol = models.CharField(db_column='Symbol', primary_key=True, max_length=10)  # Field name made lowercase.
    sector = models.CharField(db_column='Sector', max_length=40, blank=True, null=True)  # Field name made lowercase.
    location = models.CharField(db_column='Location', max_length=40, blank=True, null=True)  # Field name made lowercase.
    founded = models.IntegerField(db_column='Founded', blank=True, null=True)  # Field name made lowercase.

    class Meta:
        managed = False
        db_table = 'Company'
```

המשך – models:

```
3 usages  Lior
class Investor(models.Model):
    id = models.IntegerField(db_column='ID', primary_key=True)  # Field name made lowercase.
    name = models.CharField(db_column='Name', max_length=40, blank=True, null=True)  # Field name made lowercase.
    amount = models.FloatField(db_column='Amount', blank=True, null=True)  # Field name made lowercase.

    Lior
    class Meta:
        managed = False
        db_table = 'Investor'


2 usages  Lior *
class Stock(models.Model):
    # Field name made lowercase. The composite primary key (Symbol, tDate) found, that is not supported. The first column is selected.
    symbol = models.OneToOneField(Company, models.DO_NOTHING, db_column='Symbol', primary_key=True)
    tdate = models.DateField(db_column='tDate')  # Field name made lowercase.
    price = models.FloatField(db_column='Price', blank=True, null=True)  # Field name made lowercase.

    Lior
    class Meta:
        managed = False
        db_table = 'Stock'
        unique_together = (('symbol', 'tdate'),)


1 usage  Lior *
class Transactions(models.Model):
    # Field name made lowercase. The composite primary key (tDate, ID) found, that is not supported. The first column is selected.
    tdate = models.DateField(db_column='tDate', primary_key=True)
    id = models.ForeignKey(Investor, models.DO_NOTHING, db_column='ID')  # Field name made lowercase.
    tamount = models.IntegerField(db_column='TAmount', blank=True, null=True)  # Field name made lowercase.

    Lior
    class Meta:
        managed = False
        db_table = 'Transactions'
        unique_together = (('tdate', 'id'),)
```

:views

views.py

```python
from django.shortcuts import render
from .models import Transactions, Investor, Buying
from django.db import connection
from datetime import datetime

10 usages  Lior
def dictfetchall(cursor):
    # Returns all rows from a cursor as a dict '''
    columns = [col[0] for col in cursor.description]
    return [dict(zip(columns, row)) for row in cursor.fetchall()]

2 usages  Lior
def index(request):
    return render(request, template_name: 'index.html')

1 usage  Lior
def Query_Results(request):
    '''
    This function is used to display the results of the queries in the Query_Results.html page.
    :param request:
    :return: Query_Results.html page with the results of the queries.
    '''

    with connection.cursor() as cursor:
        cursor.execute("""SELECT DISTINCT I.Name, SOB.TotalSum
                          FROM DiversifiedInvestor DI, SumOfBuying SOB, Investor I
                          WHERE DI.ID = SOB.ID  AND DI.ID = I.ID
                          ORDER BY SOB.TotalSum DESC;
                          """)
        sql_res = dictfetchall(cursor)

        cursor.execute("""SELECT B.Symbol, I.Name, CompanyMaxBQ.MaxCompanyBQ AS Quantity
                          FROM Buying B, PopularCompany PC, Investor I, CompanyMaxBQ
                          WHERE B.Symbol = PC.Symbol AND I.ID = B.ID
                          AND B.ID = CompanyMaxBQ.ID AND B.Symbol = CompanyMaxBQ.Symbol
                          GROUP BY B.Symbol, I.Name, CompanyMaxBQ.MaxCompanyBQ
                          ORDER BY B.Symbol ASC, I.Name ASC;
                          """)
        sql_res2 = dictfetchall(cursor)

        cursor.execute("""SELECT PC.Symbol, COUNT(B.Symbol) AS TotalBuyers
                          FROM ProfitableCompany PC LEFT OUTER JOIN Buying B
                          ON PC.Symbol = B.Symbol AND PC.tDate = B.tDate
                          GROUP BY PC.Symbol
                          ORDER BY PC.Symbol ASC;
                          """)
        sql_res3 = dictfetchall(cursor)

    return render(request, template_name: 'Query_Results.html',
                  context: {'sql_res': sql_res, 'sql_res2': sql_res2, 'sql_res3': sql_res3})
```

המשך – views:

```
1 usage   Lior
def Add_Transaction(request):
    '''
    This function is used to add a new transaction to the Transactions table.
    :param request:
    :return: Add_Transaction.html page with the results of the queries.
    '''

    first_try = 0
    is_ID = 0
    is_not_Tran = 0

    with connection.cursor() as cursor:
        cursor.execute("""SELECT TOP 10 *
                          FROM Transactions
                          ORDER BY tDate DESC, ID DESC;
                          """)
        sql_res4 = dictfetchall(cursor)

    if request.method == 'POST' and request.POST:
        first_try = 1
        new_id = request.POST["ID"]

        with connection.cursor() as cursor:
            cursor.execute("""SELECT ID
                              FROM Investor
                              WHERE ID = %s
                              GROUP BY ID;
                              """, [new_id])
            row = cursor.fetchone()
            if not row: # ID not exists
                return render(request, template_name: 'Add_Transaction.html', context: {'sql_res4': sql_res4,
                                                                                        'first_try': first_try, 'is_ID': is_ID,
                                                                                        'new_id': new_id})
            else:
                is_ID = 1
                with connection.cursor() as cursor:
                    cursor.execute("""SELECT T.ID
                                      FROM Transactions T ,(SELECT MAX(tDate) AS MaxDay
                                                            FROM Stock) LastDay
                                      WHERE T.ID = %s AND T.tDate = LastDay.MaxDay
                                      GROUP BY T.ID;
                                      """, [new_id])
                    row = cursor.fetchone()
                    if row: # Transaction exists at the last day at stock table
                        return render(request, template_name: 'Add_Transaction.html',
                                      context: {'sql_res4': sql_res4,
                                       'first_try': first_try,
                                       'is_ID': is_ID,
                                       'is_not_Tran': is_not_Tran})

                    else:
                        is_not_Tran = 1
                        new_amount = int(request.POST["Transaction"])
                        with connection.cursor() as cursor:
                            cursor.execute("""SELECT MAX(tDate) AS MaxDay
                                              FROM Stock;
                                              """)
                            today = dictfetchall(cursor)
                            today = today[:1]
                        today = today[0]['MaxDay'].strftime('%Y-%m-%d')

                        with connection.cursor() as cursor:
                            cursor.execute("""
                                INSERT INTO Transactions
                                VALUES (%s, %s, %s);
                                """, [today, new_id, new_amount])
```

המשך – views:

```python
            with connection.cursor() as cursor:
                cursor.execute("""
                    UPDATE Investor
                    SET Amount = Amount + %s
                    WHERE ID = %s;
                    """, [new_amount, new_id])

            with connection.cursor() as cursor:
                cursor.execute"""SELECT TOP 10 *
                FROM Transactions
                ORDER BY tDate DESC, ID DESC;
                """)
                sql_res4 = dictfetchall(cursor)

            return render(request, template_name: 'Add_Transaction.html',
                                context: {'sql_res4': sql_res4,
                                          'first_try' : first_try,
                                          'is_ID' : is_ID,
                                          'is_not_Tran' : is_not_Tran})

    return render(request, template_name: 'Add_Transaction.html', context: {'sql_res4': sql_res4,
                                                                            'first_try': first_try})
```

```python
1 usage   Lior
def Buy_Stocks(request):
    '''
    This function is used to add a new buying to the Buying table.
    :param request:
    :return: Buy_Stocks.html page with the results of the queries.
    '''
    first_try = 0
    is_ID = 0
    is_symbol = 0
    is_large = 0
    is_not_Buy = 0

    with connection.cursor() as cursor:
        cursor.execute"""SELECT TOP 10 *
                FROM Buying
                ORDER BY tDate DESC, ID DESC, Symbol ASC;
                """)
        sql_res5 = dictfetchall(cursor)

    if request.method == 'POST' and request.POST:
        first_try = 1
        new_id = request.POST["ID"]

        with connection.cursor() as cursor:
            cursor.execute"""SELECT ID
                    FROM Investor
                    WHERE ID = %s
                    GROUP BY ID;
                    """, [new_id])
            row = cursor.fetchone()
            if row: # ID exists
                is_ID = 1

        new_symbol = request.POST["Symbol"]
```

המשך - views:

```python
        with connection.cursor() as cursor:
            cursor.execute("""SELECT S.Symbol
                              FROM Stock S, (SELECT MAX(tDate) AS MaxDay
                                             FROM Stock LastDay) LastDay
                              WHERE S.Symbol = %s AND S.tDate = LastDay.MaxDay
                              GROUP BY S.Symbol;
                              """, [new_symbol])
            row = cursor.fetchone()
            if row: # Symbol exists
                is_symbol = 1

        if (is_ID == 0) or (is_symbol == 0): # ID or Symbol not exists
            return render(request, template_name: 'Buy_Stocks.html',
                          context: {'sql_res5': sql_res5,
                          'first_try': first_try,
                          'is_ID': is_ID,
                          'is_symbol': is_symbol})

        with connection.cursor() as cursor:
            cursor.execute("""SELECT MAX(tDate) AS MaxDay
                              FROM Stock LastDay
                          """)
            StockMaxDay = dictfetchall(cursor)
            cursor.execute("""SELECT MAX(tDate) AS MaxDay
                              FROM Buying LastDay
                          """)
            BuyingMaxDay = dictfetchall(cursor)

```

```python
        # Buying table is updated to the last day at stock table
        if StockMaxDay[0]['MaxDay'] == BuyingMaxDay[0]['MaxDay']:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT B.ID
                                  FROM Buying B ,(SELECT MAX(tDate) AS MaxDay
                                                  FROM Buying) LastDay
                                  WHERE B.ID = %s AND B.Symbol = %s AND B.tDate = LastDay.MaxDay
                                  GROUP BY B.ID;
                                  """, [new_id, new_symbol])
                row = cursor.fetchone()

            if not row: # Buying not exists at the last day at stock table
                is_not_Buy = 1

        else:
            is_not_Buy = 1

        new_quantity = int(request.POST["Quantity"])

        with connection.cursor() as cursor:
            cursor.execute("""SELECT I.ID
                              FROM Investor I, Stock S ,(SELECT MAX(tDate) AS MaxDay
                                                         FROM Stock) LastDay
                              WHERE I.ID = %s AND S.Symbol = %s AND S.tDate = LastDay.MaxDay
                                    AND I.Amount >= S.Price * (%s)
                              GROUP BY I.ID;
                              """, [new_id, new_symbol, new_quantity])
            row = cursor.fetchone()

        if row: # The investor has enough money to buy the quantity of stocks he wants of the company
            is_large = 1

```

המשך – views:

```python
        # The investor doesn't have enough money or the buying exists at the last day at stock table
        if (is_large == 0) or (is_not_Buy == 0):
            return render(request, template_name: 'Buy_Stocks.html',
                          context: {'sql_res5': sql_res5,
                                    'first_try': first_try,
                                    'is_ID': is_ID,
                                    'is_symbol': is_symbol,
                                    'is_not_Buy': is_not_Buy,
                                    'is_large': is_large})

        if is_ID and is_symbol and is_not_Buy and is_large: # All the conditions are met

            today = StockMaxDay[:1]
            today = today[0]['MaxDay'].strftime('%Y-%m-%d')

            with connection.cursor() as cursor:
                cursor.execute("""
                               INSERT INTO Buying
                               VALUES (%s, %s, %s, %s);
                               """, [today, new_id, new_symbol, new_quantity])

            with connection.cursor() as cursor:
                cursor.execute("""
                           UPDATE Investor
                           SET Amount = Amount - (%s * (SELECT Price FROM Stock
                                                        WHERE Symbol = %s AND tDate = %s))
                           WHERE ID = %s;
                           """, [new_quantity, new_symbol, today, new_id])

            with connection.cursor() as cursor:
                cursor.execute"""SELECT TOP 10 *
                               FROM Buying
                               ORDER BY tDate DESC, ID DESC, Symbol ASC;
                               """)
                sql_res5 = dictfetchall(cursor)
```

```python

            return render(request, template_name: 'Buy_Stocks.html',
                          context: {'sql_res5': sql_res5,
                                    'first_try': first_try,
                                    'is_ID': is_ID,
                                    'is_symbol': is_symbol,
                                    'is_large': is_large,
                                    'is_not_Buy': is_not_Buy})

    return render(request, template_name: 'Buy_Stocks.html', context: {'sql_res5': sql_res5,
                                                                        'first_try': first_try})
```