Karthik Potturi's Task

Developer	@Karthik Potturi
Approver	@Karthik Potturi
Informed Stakeholders	ADDX
Status	DONE

3 Tasks Assignment

Data Sourcing Activity

Python Code

```python
import requests, time
import pandas as pd

def fetch_all_records(api_url, max_retries = 3, retry_delay = 1):
    headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch
    records = []
    offset = 0
    limit = 100
    retries = 0
    while retries <= max_retries:
        params = {
            'resource_id' : '3a5b732e-9490-4629-a398-d0c414204ee0',
            'limit': limit,
            'offset':offset,
            'sort': 'end_of_week desc'
        }
        response = requests.get(api_url, headers=headers, params=params)
        if response.status_code == 200:
            data = response.json()
            records.extend(data['result']['records'])
            print(len(data['result']['records']))
            if len(data['result']['records']) < limit:
                break
            offset = offset + limit
        else:
            print("Failed to fetch the data. Status Code {}".format(response.status_code))
            if retries < max_retries:
                retries += 1
                print('Retrying in {} seconds'.format(retry_delay))
                time.sleep(retry_delay)
            else:
                print('Max attempts reached. Unable to fetch data')
                break
    return records

if __name__ == '__main__':
    api_url = "https://eservices.mas.gov.sg/api/action/datastore/search.json"
```

```
    all_records = fetch_all_records(api_url)

    data_dict = {record['end_of_week']:record for record in all_records}
    #print("Total records {}".format(len(data_dict)))
    df = pd.DataFrame.from_dict(data_dict)
    df.to_csv(r'exchange_rate_data.csv', index=True, header=False)
```

🎯 Data Modelling Activity

Database: MSSQL Express 2019

MSSQL Management Studio

No Of Tables: 5

Stored Procedure: 1 (Optional for demo purpose use store procedure instead of standard query)

Functions : 2

Models

For this assessment I am using MSSQL Express 2019 . Below are the list of tables and their table structures

Table: cf_customer

Name	DataType	Allow Nulls	Remarks
custid	bigint	N	Primary Key
email	nvarchar(100)	Y	
first_name	nvarchar(50)	Y	
last_name	nvarchar(50)	Y	
password	nvarchar(250)	Y	
contactno	nvarchar(20)	Y	
pwd_failed_count	int	N	
guid	uniqueidentifier	Y	System auto generated when user click on reset password
pwd_req_date	datetime	Y	Reset password req date, later use for validation for link expiry
islocked	bit	N	If user x number of failed count, this flag will be updated to 'Y' , So user will not login.
device	nvarchar(50)	Y	Last access device will be updated

ipaddress	nvarchar(50)	Y	Last access customer IP address
status	nvarchar(50)	Y	Pending / Review for Approval / Approved / Rejected
remarks	nvarchar(MAX)	Y	if rejected, admin update remarks
review_sys_userid	int	Y	reviewer admin system userid
approved_date	datetime	Y	
last_updated	datetime	Y	System Date Time
create_date	datetime	Y	System Date Time

Table: cf_login_history

Name	DataType	Allow Nulls	Remarks
id	bigint	N	Primary Key
cust_id	bigint	N	Foreign Key with cf_customer table
channel	nvarchar(50)	Y	eg: User agent details like iPhone Safari …
ipaddress	nvarchar(50)	Y	customer ip address
login_status	char(1)	Y	Y / F (Success / Failed)
create_date	datetime	Y	System Date Time

Table: cf_product

Name	DataType	Allow Nulls	Remarks
pid	bigint	N	Primary Key
name	nvarchar(50)	Y	
price	decimal(10,2)	N	
qty	int	N	
sale_start_date	datetime	Y	
sale_end_date	datetime	Y	
status	char(1)	Y	
last_updated	datetime	Y	System Date Time
create_date	datetime	Y	System Date Time

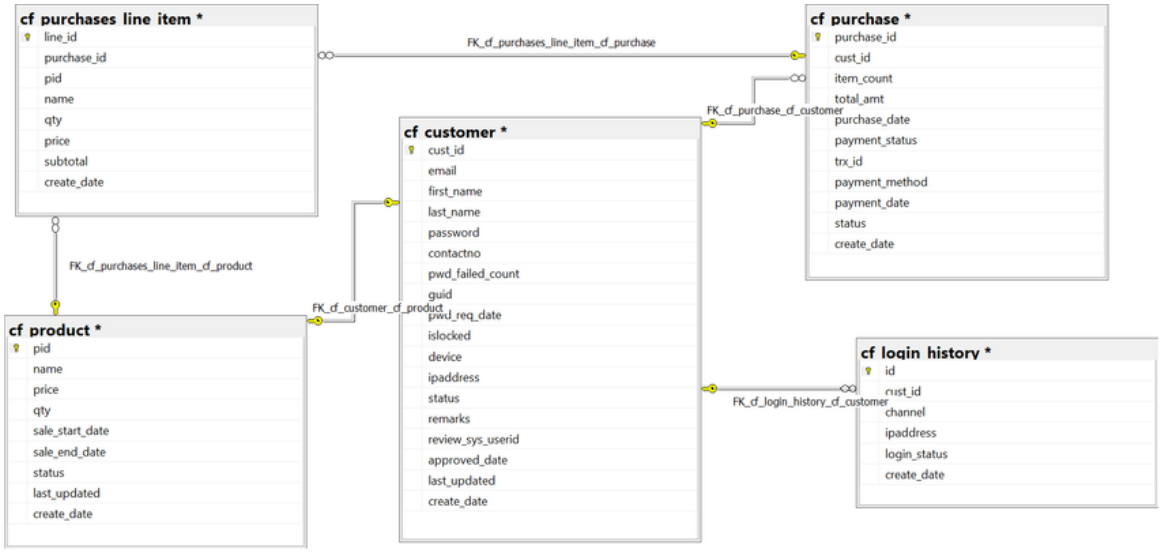
Table: cf_purchase

Name	DataType	Allow Nulls	Remarks
purchase_id	bigint	N	Primary Key
cust_id	bigint	N	Foreign Key with cf_customerTable
item_count	int	N	
total_amt	decimal(10,2)	N	
purchase_date	datetime	N	
payment_status	nvarchar(50)	N	
trx_id	uniqueidentifier	N	System auto generated upon checkout
payment_method	nvarchar(50)	Y	
payment_date	datetime	Y	
status	nvarchar(50)	N	
create_date	datetime	N	System Date Time

Table: cf_purchase_line_item

Name	DataType	Allow Nulls	Remarks
line_id	bigint	N	Primary Key
purchase_id	bigint	N	Foreign Key with cf_purchase table
pid	bigint	N	Can not be foregin key, product can be delete after x days if required, but user transaction can not be delete and can not reference, product name, price will capture from product table when time of checkout
name	nvarchar(50)	Y	
qty	int	N	
price	decimal(10,2)	N	
subtotal	decimal(10,2)	N	Auto calculate qty x price (assume no discounts, tax for this task)
create_date	datetime	Y	System DateTime

ER Diagram


cf purchases line item *
line_id
purchase_id
pid
name
qty
price
subtotal
create_date
FK_cf_purchases_line_item_cf_purchase
cf purchase *
purchase_id
cust_id
item_count
total_amt
purchase_date
payment_status
trx_id
payment_method
payment_date
status
create_date
FK_cf_purchase_cf_customer
cf customer *
cust_id
email
first_name
last_name
password
contactno
pwd_failed_count
guid
pwd_req_date
islocked
device
ipaddress
status
remarks
review_sys_userid
approved_date
last_updated
create_date
FK_cf_purchases_line_item_cf_product
FK_cf_customer_cf_product
cf product *
pid
name
price
qty
sale_start_date
sale_end_date
status
last_updated
create_date
cf login history *
id
cust_id
channel
ipaddress
login_status
create_date
FK_cf_login_history_cf_customer

Relationships

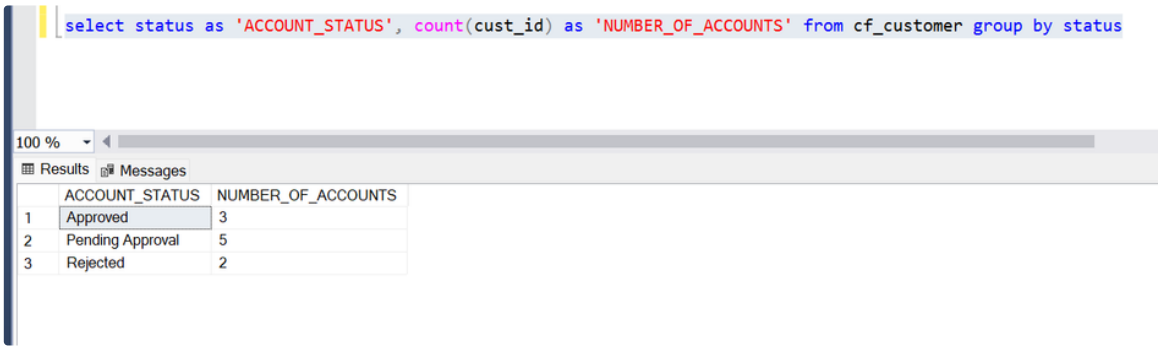
1. **cf_customer** and **cf_login_history**:
   a. Relationship: One-to-Many (1:N)
   b. Explanation: One customer can have multiple login history records (one-to-many relationship based on cust_id).
2. **cf_customer** and **cf_purchase**:
   a. Relationship: One-to-Many (1:N)
   b. Explanation: One customer can have multiple purchase records (one-to-many relationship based on cust_id).
3. **cf_purchase** and **cf_purchases_line_item**:
   a. Relationship: One-to-Many (1:N)
   b. Explanation: One purchase can have multiple line items (one-to-many relationship based on purchase_id).
4. **cf_customer** and **cf_product**:
   a. Relationship: Many-to-Many (M:N) through a junction table
   b. Explanation: A customer can purchase multiple products, and a product can be purchased by multiple customers. This requires a junction table (not explicitly mentioned) to represent this many-to-many relationship.

🎯 Data Manipulation Activity

Below are the SQL Queries for questions in the order of a-e

ℹ Answer for Question 'a'

```
select status as 'ACCOUNT_STATUS', count(cust_id) as 'NUMBER_OF_ACCOUNTS' from cf_customer group by status
```

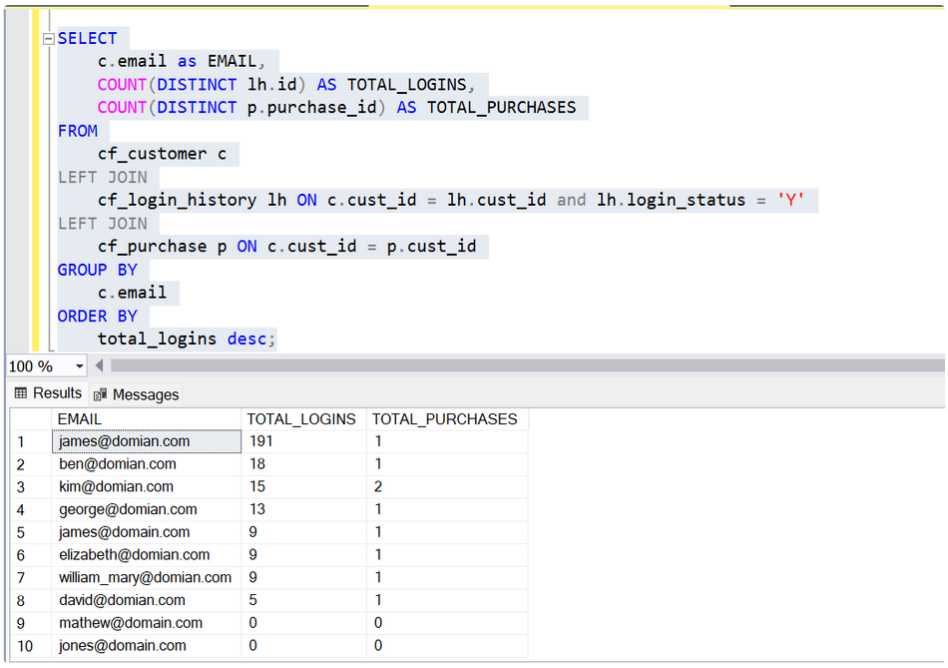


Screenshot 'a'

Answer for Question 'b'

```sql
/*--------------*/
Approach - 1
/*--------------*/
SELECT
    c.email as EMAIL,
    COUNT(DISTINCT lh.id) AS TOTAL_LOGINS,
    COUNT(DISTINCT p.purchase_id) AS TOTAL_PURCHASES
FROM
    cf_customer c
LEFT JOIN
    cf_login_history lh ON c.cust_id = lh.cust_id and lh.login_status = 'Y'
LEFT JOIN
    cf_purchase p ON c.cust_id = p.cust_id
GROUP BY
    c.email
ORDER BY
    total_logins desc;

/*--------------*/
Approach - 2
/*--------------*/
dbo.getUserLoginCount and dbo.getUserPurchaseCount are Functions that return the counts
SELECT
    email as EMAIL,
    dbo.getUserLoginCount(cust_id) as TOTAL_LOGINS,
    dbo.getUserPurchaseCount(cust_id) as TOTAL_LOGINS
from cf_customer
```



```sql
SELECT
    c.email as EMAIL,
    COUNT(DISTINCT lh.id) AS TOTAL_LOGINS,
    COUNT(DISTINCT p.purchase_id) AS TOTAL_PURCHASES
FROM
    cf_customer c
LEFT JOIN
    cf_login_history lh ON c.cust_id = lh.cust_id and lh.login_status = 'Y'
LEFT JOIN
    cf_purchase p ON c.cust_id = p.cust_id
GROUP BY
    c.email
ORDER BY
    total_logins desc;
```

100 %

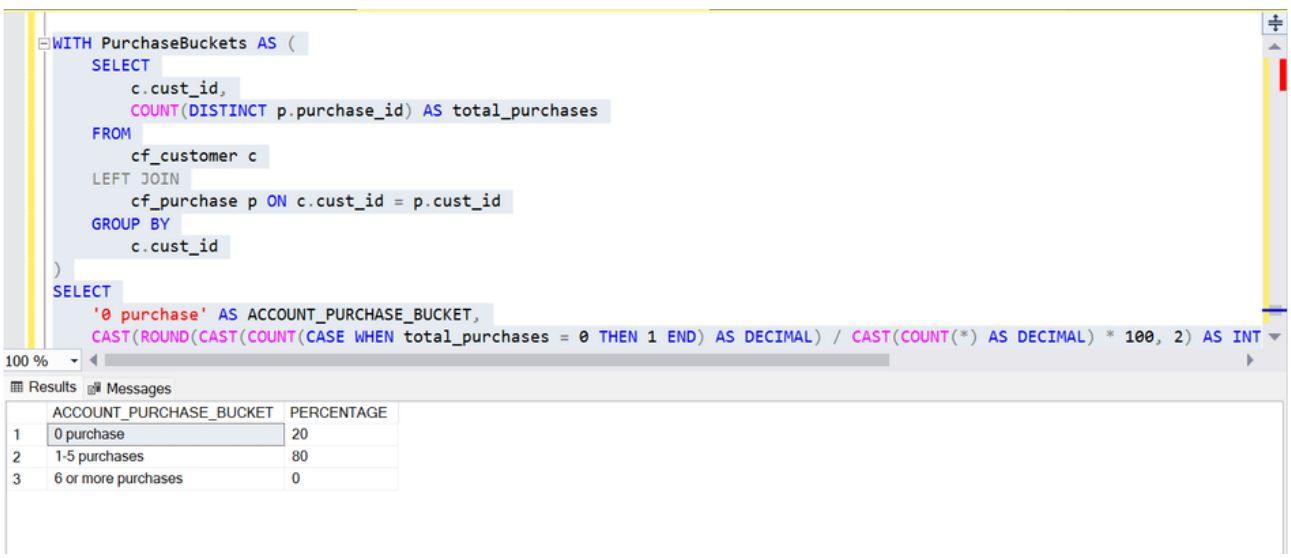
Results | Messages

	EMAIL	TOTAL_LOGINS	TOTAL_PURCHASES
1	james@domian.com	191	1
2	ben@domian.com	18	1
3	kim@domian.com	15	2
4	george@domian.com	13	1
5	james@domain.com	9	1
6	elizabeth@domian.com	9	1
7	william_mary@domian.com	9	1
8	david@domian.com	5	1
9	mathew@domain.com	0	0
10	jones@domain.com	0	0

Screenshot 'b'

Answer for Question 'c'

```sql
WITH PurchaseBuckets AS (
    SELECT
        c.cust_id,
        COUNT(DISTINCT p.purchase_id) AS total_purchases
    FROM
        cf_customer c
    LEFT JOIN
        cf_purchase p ON c.cust_id = p.cust_id
    GROUP BY
        c.cust_id
)
SELECT
    '0 purchase' AS ACCOUNT_PURCHASE_BUCKET,
    CAST(ROUND(CAST(COUNT(CASE WHEN total_purchases = 0 THEN 1 END) AS DECIMAL) / CAST(COUNT(*) AS DECIMAL) * 10
FROM
    PurchaseBuckets
UNION ALL
SELECT
    '1-5 purchases' AS ACCOUNT_PURCHASE_BUCKET,
    CAST(ROUND(CAST(COUNT(CASE WHEN total_purchases BETWEEN 1 AND 5 THEN 1 END) AS DECIMAL) / CAST(COUNT(*) AS D
FROM
    PurchaseBuckets
UNION ALL
SELECT
    '6 or more purchases' AS ACCOUNT_PURCHASE_BUCKET,
    CAST(ROUND(CAST(COUNT(CASE WHEN total_purchases >= 6 THEN 1 END) AS DECIMAL) / CAST(COUNT(*) AS DECIMAL) * 1
FROM
    PurchaseBuckets;
```

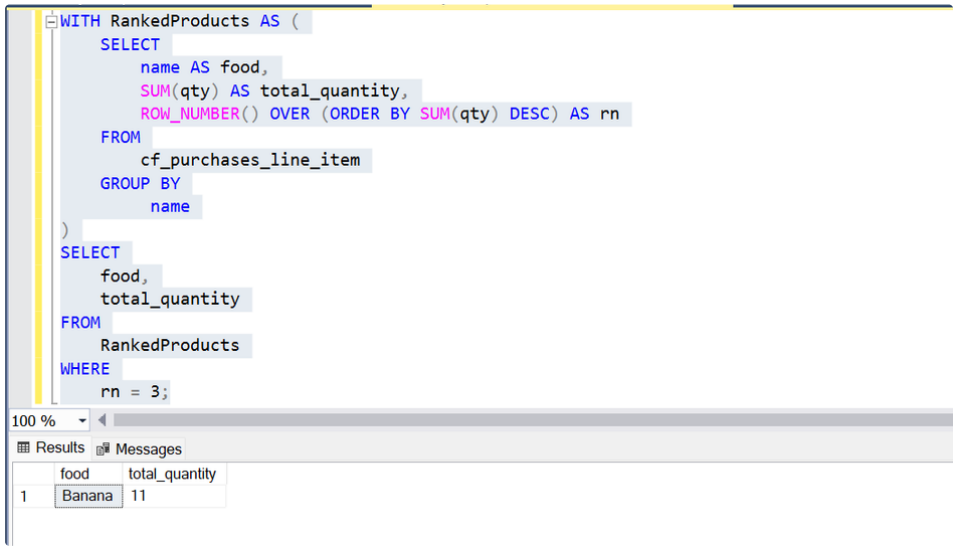


Screenshot 'c'

Answer for Question 'd'

```sql
WITH RankedProducts AS (
    SELECT
        name AS food,
        SUM(qty) AS total_quantity,
        ROW_NUMBER() OVER (ORDER BY SUM(qty) DESC) AS rn
    FROM
```

```
        cf_purchases_line_item
    GROUP BY
         name
)
SELECT
    FOOD,
    TOTAL_QUANTITY
FROM
    RankedProducts
WHERE
    rn = 3;
```



Screenshot 'd'

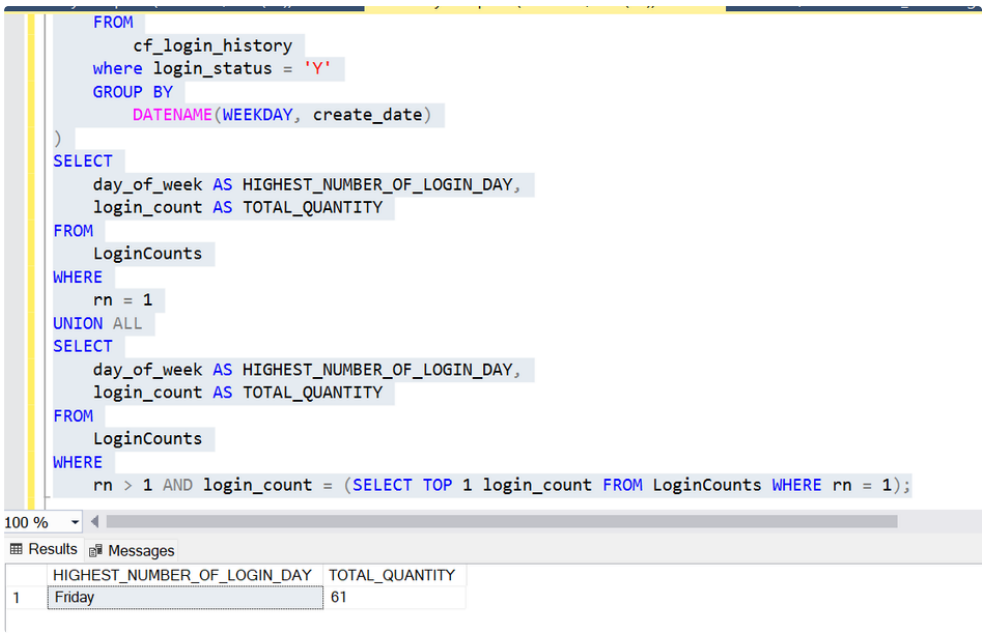
Answer for Question 'e'

```
WITH LoginCounts AS (
    SELECT
        DATENAME(WEEKDAY, create_date) AS day_of_week,
        COUNT(*) AS login_count,
        ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC) AS rn
    FROM
        cf_login_history
    where login_status = 'Y'
    GROUP BY
        DATENAME(WEEKDAY, create_date)
)
SELECT
    day_of_week AS HIGHEST_NUMBER_OF_LOGIN_DAY,
    login_count AS TOTAL_QUANTITY
FROM
    LoginCounts
WHERE
    rn = 1

UNION ALL

SELECT
    day_of_week AS HIGHEST_NUMBER_OF_LOGIN_DAY,
    login_count AS TOTAL_QUANTITY
```

```
FROM
    LoginCounts
WHERE
    rn > 1 AND login_count = (SELECT TOP 1 login_count FROM LoginCounts WHERE rn = 1);
```



Screenshot 'e'