

## Karthik Potturi's Task

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

 3 Tasks Assignment

## Data Sourcing Activity

 Python Code

```
1 import requests, time
2 import pandas as pd
3
4 def fetch_all_records(api_url, max_retries = 3, retry_delay = 1):
5     headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Ch
6     records = []
7     offset = 0
8     limit = 100
9     retries = 0
10    while retries <= max_retries:
11        params = {
12            'resource_id' : '3a5b732e-9490-4629-a398-d0c414204ee0',
13            'limit': limit,
14            'offset': offset,
15            'sort': 'end_of_week desc'
16        }
17        response = requests.get(api_url, headers=headers, params=params)
18        if response.status_code == 200:
19            data = response.json()
20            records.extend(data['result']['records'])
21            print(len(data['result']['records']))
22            if len(data['result']['records']) < limit:
23                break
24            offset = offset + limit
25        else:
26            print("Failed to fetch the data. Status Code {}".format(response.status_code))
27            if retries < max_retries:
28                retries += 1
29                print('Retrying in {} seconds'.format(retry_delay))
30                time.sleep(retry_delay)
31            else:
32                print('Max attempts reached. Unable to fetch data')
33                break
34    return records
35
36 if __name__ == '__main__':
37     api_url = "https://eservices.mas.gov.sg/api/action/datastore/search.json"
```

```

38 all_records = fetch_all_records(api_url)
39
40 data_dict = {record['end_of_week']:record for record in all_records}
41 #print("Total records {}".format(len(data_dict)))
42 df = pd.DataFrame.from_dict(data_dict)
43 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	Data Type	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	Data Type	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

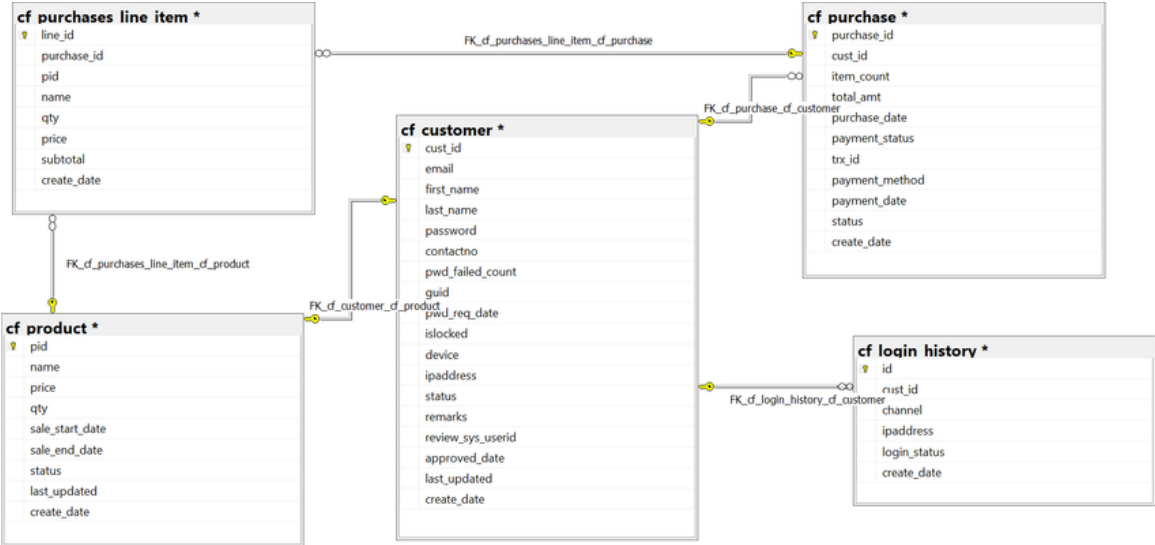
**Table: cf\_purchase**

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




## 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

 Answer for Question 'a'

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

 Answer for Question 'b'

```
1 /*-----*/
2 Approach - 1
3 /*-----*/
4 SELECT
5     c.email,
6     COUNT(DISTINCT lh.id) AS total_logins,
7     COUNT(DISTINCT p.purchase_id) AS total_purchases
8 FROM
9     cf_customer c
```

```

10 LEFT JOIN
11     cf_login_history lh ON c.cust_id = lh.cust_id and lh.login_status = 'Y'
12 LEFT JOIN
13     cf_purchase p ON c.cust_id = p.cust_id
14 GROUP BY
15     c.email
16 ORDER BY
17     total_logins desc;
18
19 /*-----*/
20 Approach - 2
21 /*-----*/
22 dbo.getUserLoginCount and dbo.getUserPurchaseCount are Functions that return the counts
23 SELECT
24     email,
25     dbo.getUserLoginCount(cust_id) as Total_Logins,
26     dbo.getUserPurchaseCount(cust_id) as Total_Purchase
27 from cf_customer

```

#### Answer for Question 'c'

```

1 WITH PurchaseBuckets AS (
2     SELECT
3         c.cust_id,
4         COUNT(DISTINCT p.purchase_id) AS total_purchases
5     FROM
6         cf_customer c
7     LEFT JOIN
8         cf_purchase p ON c.cust_id = p.cust_id
9     GROUP BY
10        c.cust_id
11 )
12 SELECT
13     '0 purchase' AS bucket,
14     CAST(ROUND(CAST(COUNT(CASE WHEN total_purchases = 0 THEN 1 END) AS DECIMAL) / CAST(COUNT(*) AS DECIMAL) * 100, 2) AS DECIMAL) AS percentage
15 FROM
16     PurchaseBuckets
17 UNION ALL
18 SELECT
19     '1-5 purchases' AS bucket,
20     CAST(ROUND(CAST(COUNT(CASE WHEN total_purchases BETWEEN 1 AND 5 THEN 1 END) AS DECIMAL) / CAST(COUNT(*) AS DECIMAL) * 100, 2) AS DECIMAL) AS percentage
21 FROM
22     PurchaseBuckets
23 UNION ALL
24 SELECT
25     '6 or more purchases' AS bucket,
26     CAST(ROUND(CAST(COUNT(CASE WHEN total_purchases >= 6 THEN 1 END) AS DECIMAL) / CAST(COUNT(*) AS DECIMAL) * 100, 2) AS DECIMAL) AS percentage
27 FROM
28     PurchaseBuckets;

```

#### Answer for Question 'd'

```

1 /*-----*/
2 Approach - 1
3 /*-----*/
4 WITH RankedProducts AS (
5     SELECT

```

```

6         name AS food,
7         SUM(qty) AS total_quantity,
8         ROW_NUMBER() OVER (ORDER BY SUM(qty) DESC) AS rn
9     FROM
10        cf_purchases_line_item
11    GROUP BY
12        name
13 )
14 SELECT
15     food,
16     total_quantity
17 FROM
18     RankedProducts
19 WHERE
20     rn = 3;
21
22 /*-----*/
23 Approach - 2
24 /*-----*/
25 SELECT
26     b.name,
27     SUM(a.qty)TotalQty
28 FROM
29     cf_purchases_line_item a
30 JOIN
31     cf_product b
32 ON a.pid = b.pid
33 GROUP BY b.name
34 ORDER BY SUM(a.qty) DESC
35 OFFSET 2 ROWS
36 FETCH NEXT 1 ROW ONLY

```

#### Answer for Question 'e'

```

1 WITH LoginCounts AS (
2     SELECT
3         DATENAME(WEEKDAY, create_date) AS day_of_week,
4         COUNT(*) AS login_count,
5         ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC) AS rn
6     FROM
7         cf_login_history
8     where login_status = 'Y'
9     GROUP BY
10        DATENAME(WEEKDAY, create_date)
11 )
12 SELECT
13     day_of_week,
14     login_count AS total_quantity
15 FROM
16     LoginCounts
17 WHERE
18     rn = 1
19
20 UNION ALL
21
22 SELECT
23     day_of_week,
24     login_count AS total_quantity

```



```
25 FROM
26     LoginCounts
27 WHERE
28     rn > 1 AND login_count = (SELECT TOP 1 login_count FROM LoginCounts WHERE rn = 1);
```