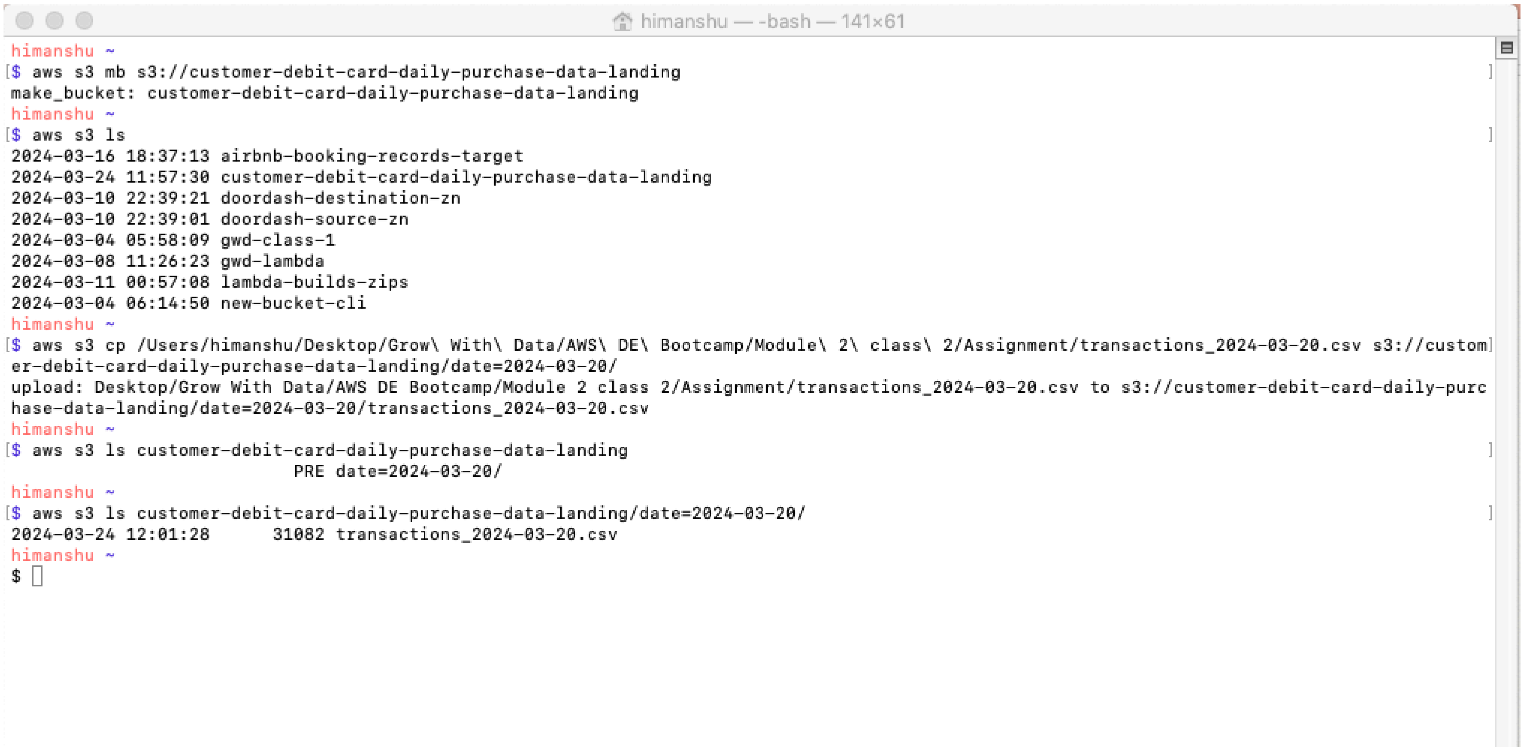
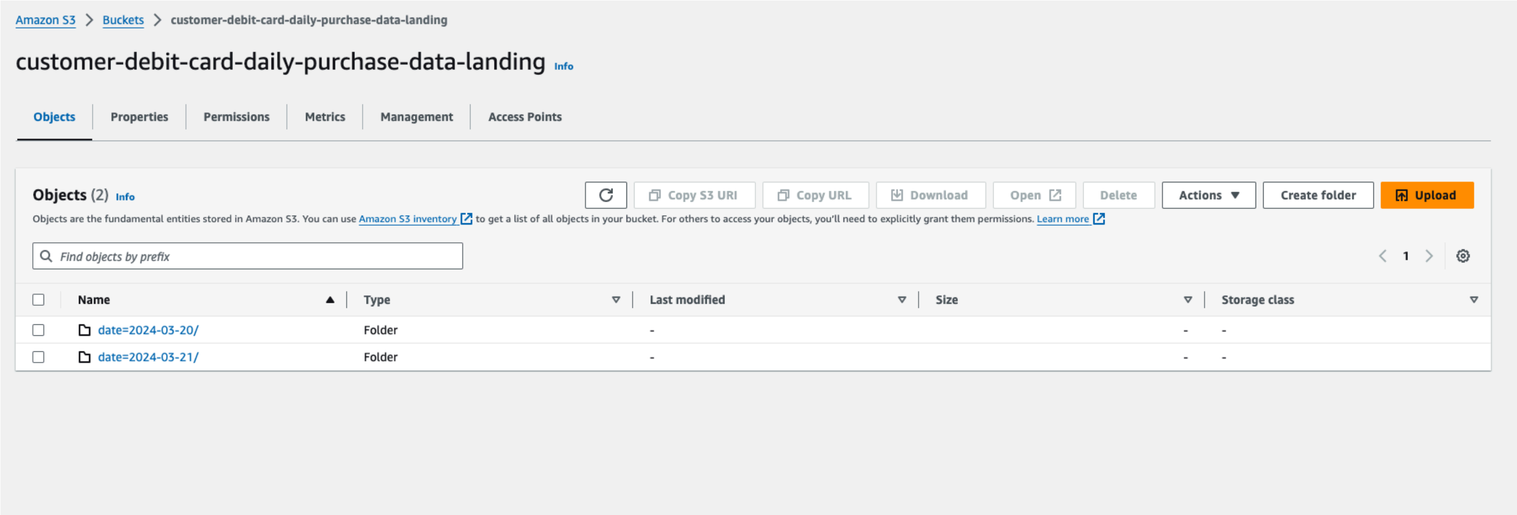


```
himanshu ~
[$ aws s3 mb s3://customer-debit-card-daily-purchase-data-landing                                                                              ]
make_bucket: customer-debit-card-daily-purchase-data-landing
himanshu ~
[$ aws s3 ls                                                                                                                                   ]
2024-03-16 18:37:13 airbnb-booking-records-target
2024-03-24 11:57:30 customer-debit-card-daily-purchase-data-landing
2024-03-10 22:39:21 doordash-destination-zn
2024-03-10 22:39:01 doordash-source-zn
2024-03-04 05:58:09 gwd-class-1
2024-03-08 11:26:23 gwd-lambda
2024-03-11 00:57:08 lambda-builds-zips
2024-03-04 06:14:50 new-bucket-cli
himanshu ~
[$ aws s3 cp /Users/himanshu/Desktop/Grow\ With\ Data/AWS\ DE\ Bootcamp/Module\ 2\ class\ 2/Assignment/transactions_2024-03-20.csv s3://custom]
er-debit-card-daily-purchase-data-landing/date=2024-03-20/
upload: Desktop/Grow With Data/AWS DE Bootcamp/Module 2 class 2/Assignment/transactions_2024-03-20.csv to s3://customer-debit-card-daily-purc
hase-data-landing/date=2024-03-20/transactions_2024-03-20.csv
himanshu ~
[$ aws s3 ls customer-debit-card-daily-purchase-data-landing                                                                                   ]
                           PRE date=2024-03-20/
himanshu ~
[$ aws s3 ls customer-debit-card-daily-purchase-data-landing/date=2024-03-20/                                                                  ]
2024-03-24 12:01:28      31082 transactions_2024-03-20.csv
himanshu ~
$ 
```



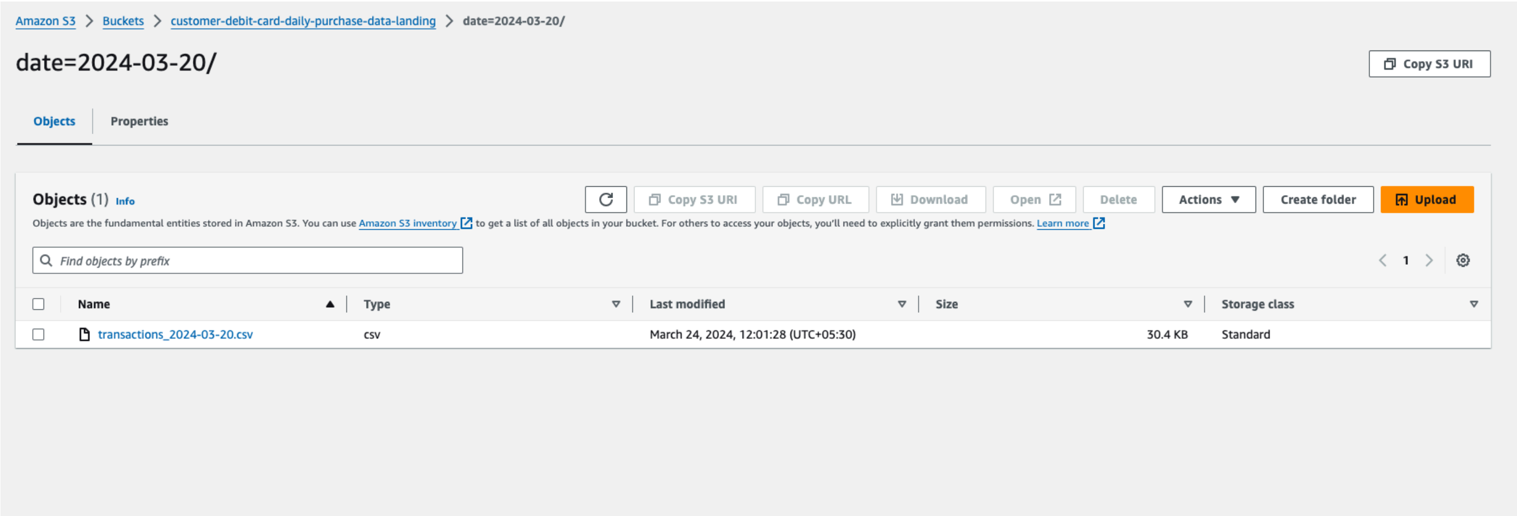
Amazon S3 > Buckets > customer-debit-card-daily-purchase-data-landing

customer-debit-card-daily-purchase-data-landing Info

Objects (2) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type	Last modified	Size	Storage class
□	date=2024-03-20/	Folder	-	-	-
□	date=2024-03-21/	Folder	-	-	-



Amazon S3 > Buckets > customer-debit-card-daily-purchase-data-landing > date=2024-03-20/

date=2024-03-20/

Objects (1) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type	Last modified	Size	Storage class
□	transactions_2024-03-20.csv	csv	March 24, 2024, 12:01:28 (UTC+05:30)	30.4 KB	Standard

As shown in the screenshots,hive style data partition subfolders have been created, which follows partition based on date.

The SQL statement creates a table named `daily_aggregation`, consisting of columns `customer_id` (VARCHAR, max length 10), `debit_card_number` (VARCHAR, max length 255), `bank_name` (VARCHAR, max length 255), and `total_amount_spend` (FLOAT), with `customer_id` set as the primary key.

Now coming to the Glue job part.

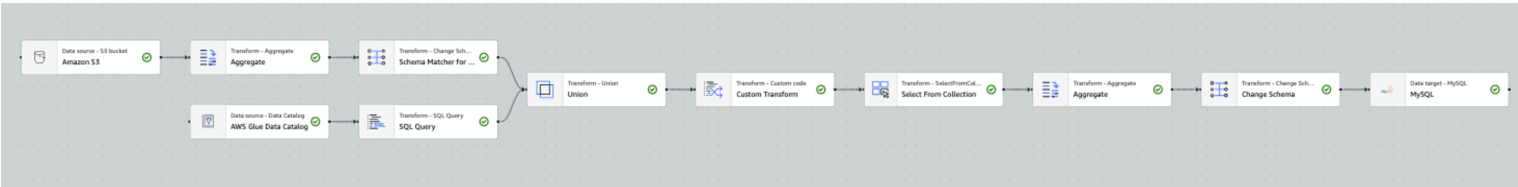
This Glue ETL job script performs the following tasks:

1. Reads data from an Amazon S3 source table named `s3_input_customer_debit_card_daily_purchase_data_landing` in the `transaction-metadata-db` database.
2. Reads data from an AWS Glue Data Catalog source table named `debit_card_spends_daily_aggregation` in the `transaction-metadata-db` database.
3. Aggregates data from the S3 source table by grouping it based on `customer_id`, `debit_card_number`, and `bank_name`, and calculates the sum of the `amount_spend`.
4. Executes an SQL query to select data from a data source named `myDataSource`.
5. Matches the schema for the Union operation, mapping the columns accordingly.
6. Unions the data from the SQL query result and the aggregated data.
7. Performs a custom transformation, which truncates the `daily_aggregation` table in a MySQL database.
8. Selects data from the resulting collection.
9. Aggregates the selected data by grouping it based on `customer_id`, `debit_card_number`, and `bank_name`, and calculates the sum of the `total_amount_spend`.
10. Changes the schema of the aggregated data.
11. Writes the final aggregated data to a table named `debit_card_spends_daily_aggregation` in the `transaction-metadata-db` database.
12. Commits the job.

Overall, this ETL job script processes and aggregates data from multiple sources, performs transformations, and writes the final result to a destination table in a MySQL database.

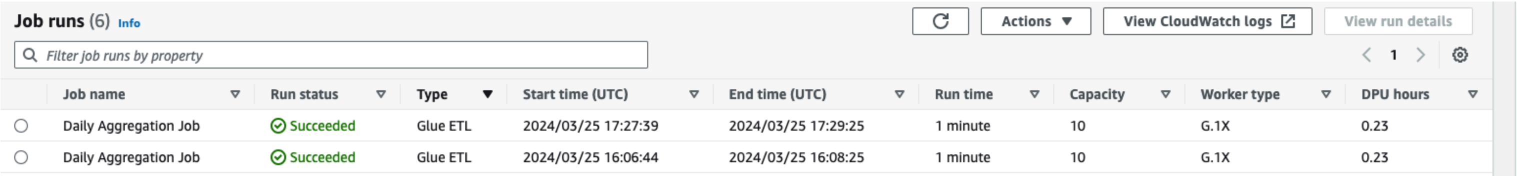
For handling incremental data ingestion, here first the data from S3 location is crawled and then from the table, then the union happens for both the tables then the custom transform function truncates the table in rds instance. Then aggregation happens all over again. Then the final result is ingested inside the truncated table in MySQL RDS instance.

GLUE ETL JOB



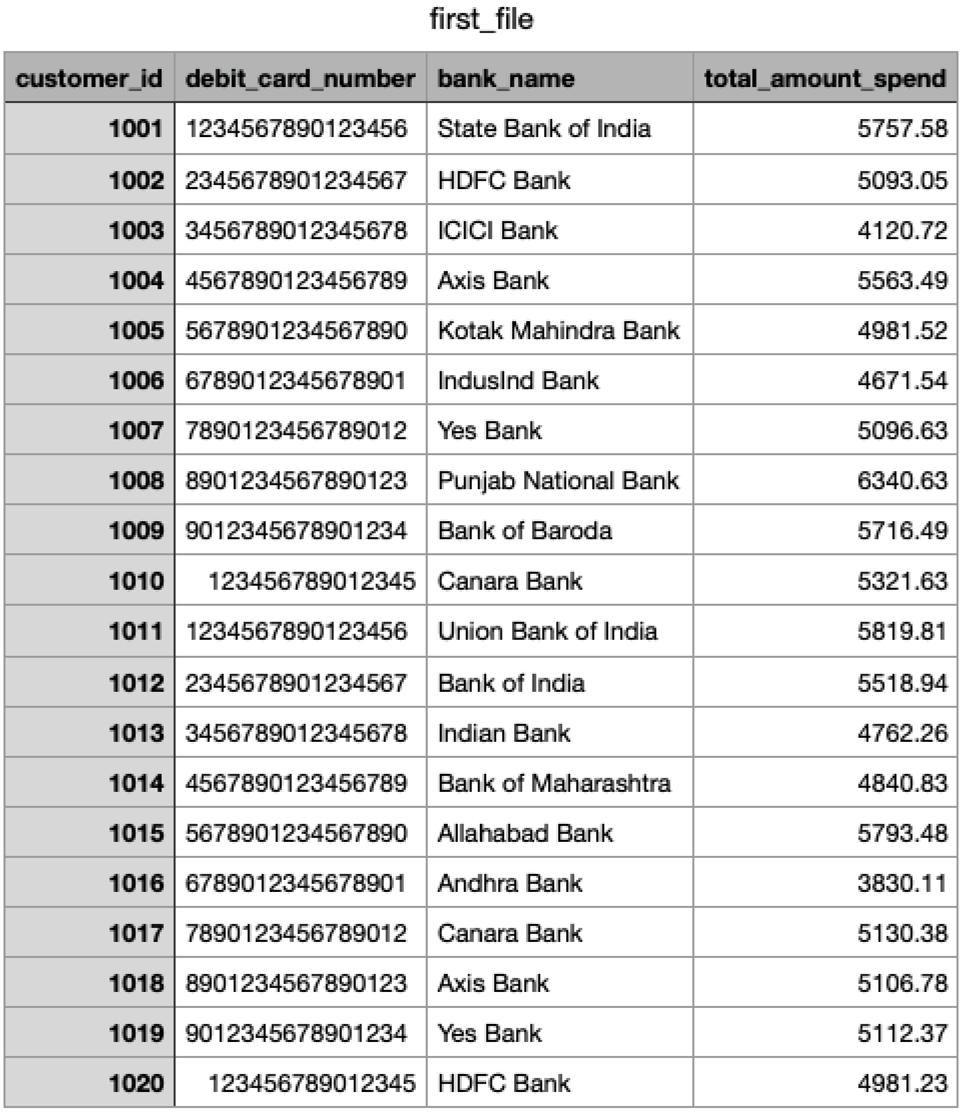
Reflection on Assignment:

The major challenge here I faced was related to truncating the data, I used the customTransformation function and then used select from collection function so that it can be used correctly.



	Job name	Run status	Type	Start time (UTC)	End time (UTC)	Run time	Capacity	Worker type	DPU hours
○	Daily Aggregation Job	✓ Succeeded	Glue ETL	2024/03/25 17:27:39	2024/03/25 17:29:25	1 minute	10	G.1X	0.23
○	Daily Aggregation Job	✓ Succeeded	Glue ETL	2024/03/25 16:06:44	2024/03/25 16:08:25	1 minute	10	G.1X	0.23

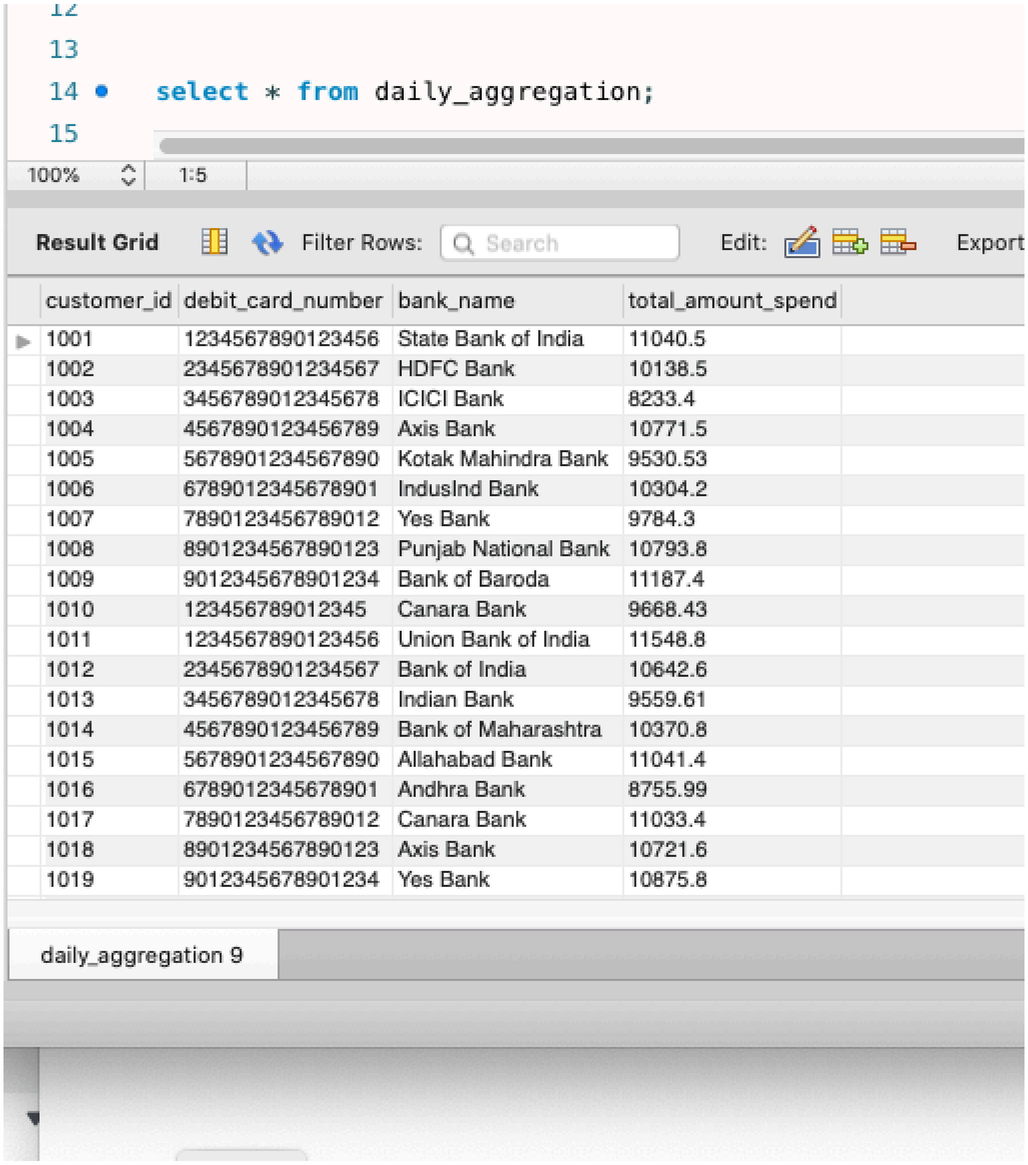
First Job execution result:



first_file

customer_id	debit_card_number	bank_name	total_amount_spend
1001	1234567890123456	State Bank of India	5757.58
1002	2345678901234567	HDFC Bank	5093.05
1003	3456789012345678	ICICI Bank	4120.72
1004	4567890123456789	Axis Bank	5563.49
1005	5678901234567890	Kotak Mahindra Bank	4981.52
1006	6789012345678901	IndusInd Bank	4671.54
1007	7890123456789012	Yes Bank	5096.63
1008	8901234567890123	Punjab National Bank	6340.63
1009	9012345678901234	Bank of Baroda	5716.49
1010	123456789012345	Canara Bank	5321.63
1011	1234567890123456	Union Bank of India	5819.81
1012	2345678901234567	Bank of India	5518.94
1013	3456789012345678	Indian Bank	4762.26
1014	4567890123456789	Bank of Maharashtra	4840.83
1015	5678901234567890	Allahabad Bank	5793.48
1016	6789012345678901	Andhra Bank	3830.11
1017	7890123456789012	Canara Bank	5130.38
1018	8901234567890123	Axis Bank	5106.78
1019	9012345678901234	Yes Bank	5112.37
1020	123456789012345	HDFC Bank	4981.23

Second job execution result:



```
select * from daily_aggregation;
```

customer_id	debit_card_number	bank_name	total_amount_spend
1001	1234567890123456	State Bank of India	11040.5
1002	2345678901234567	HDFC Bank	10138.5
1003	3456789012345678	ICICI Bank	8233.4
1004	4567890123456789	Axis Bank	10771.5
1005	5678901234567890	Kotak Mahindra Bank	9530.53
1006	6789012345678901	IndusInd Bank	10304.2
1007	7890123456789012	Yes Bank	9784.3
1008	8901234567890123	Punjab National Bank	10793.8
1009	9012345678901234	Bank of Baroda	11187.4
1010	123456789012345	Canara Bank	9668.43
1011	1234567890123456	Union Bank of India	11548.8
1012	2345678901234567	Bank of India	10642.6
1013	3456789012345678	Indian Bank	9559.61
1014	4567890123456789	Bank of Maharashtra	10370.8
1015	5678901234567890	Allahabad Bank	11041.4
1016	6789012345678901	Andhra Bank	8755.99
1017	7890123456789012	Canara Bank	11033.4
1018	8901234567890123	Axis Bank	10721.6
1019	9012345678901234	Yes Bank	10875.8