OPIM 5272: Data Management and Business Process Modeling

<u>Project Phase 3</u>

Team 13:

Abhinav Dubey
Bhavya Bansal
Mayank Patidar
Priyanka Patel

University of Connecticut

MS in Business Analytics and Project Management

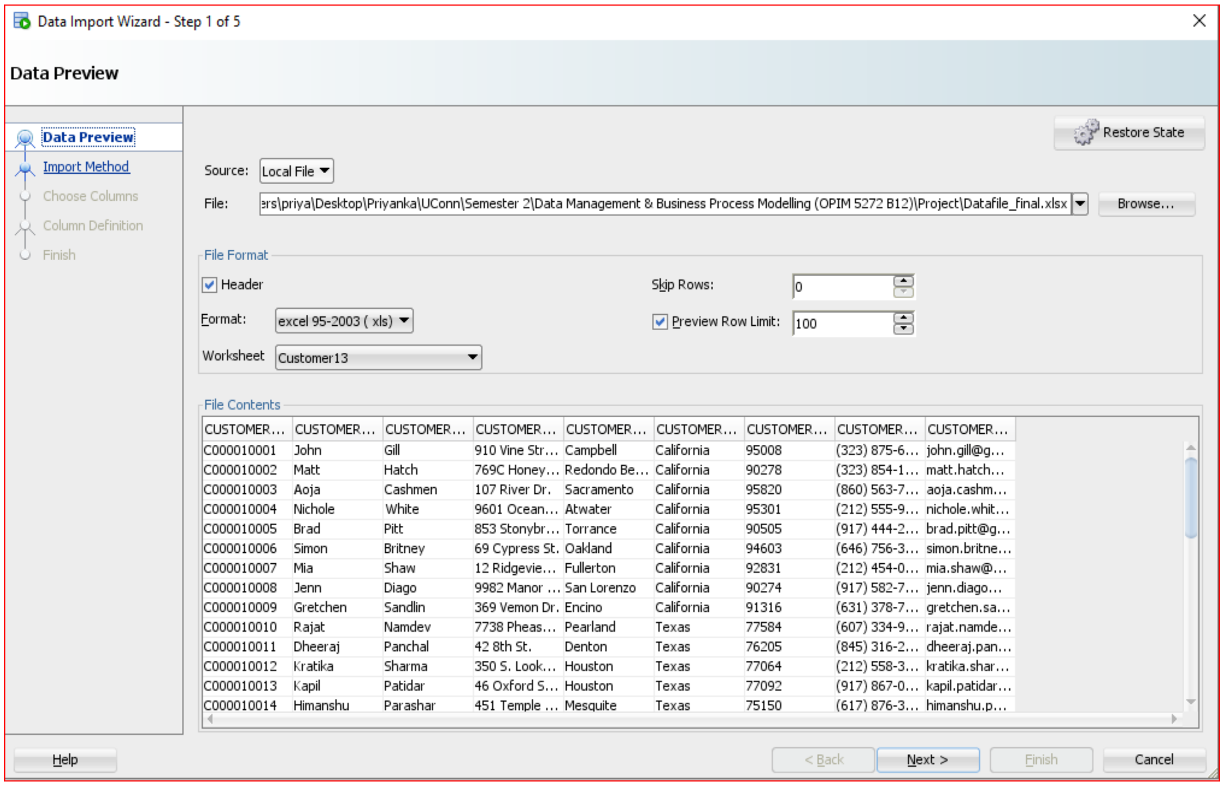
Table of Contents

<u>Loading the records</u>

In order to load the records in the tables created, we imported the same through an excel file. We had 15 different tables and to ensure the authenticity and proper relatability in the data, we first filled in all the data in an excel file. Following are the steps that we followed to import the data in Oracle SQL Developer:

Step 1: Right click on the table name and select ‘import data’.

Step 2: A pop-up headed “Data Import Wizard” will open. Upload the excel file having the data. Select the appropriate worksheet (the table that needs to be loaded). Click Next.



Source: Local File

File: ers\priya\Desktop\Priyanka\UConn\Semester 2\Data Management & Business Process Modelling (OPIM 5272 B12)\Project\Datafile_final.xlsx

File Format

☑ Header

Skip Rows: 0

Format: excel 95-2003 (xls)

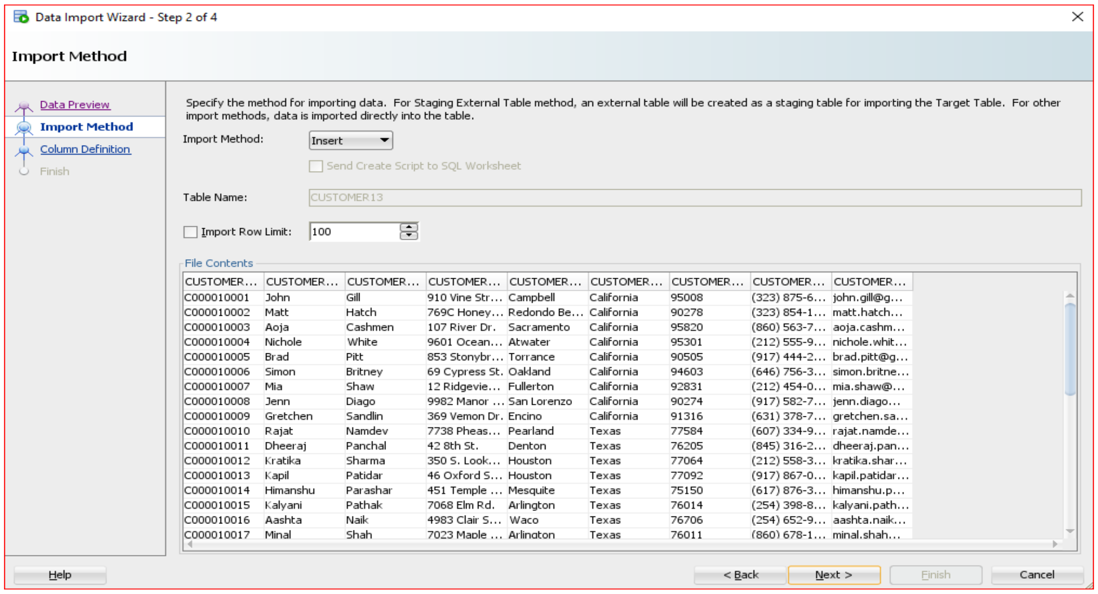
☑ Preview Row Limit: 100

Worksheet Customer13

File Contents

CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...
C000010001	John	Gill	910 Vine Str...	Campbell	California	95008	(323) 875-6...	john.gill@g...
C000010002	Matt	Hatch	769C Honey...	Redondo Be...	California	90278	(323) 854-1...	matt.hatch...
C000010003	Aoja	Cashmen	107 River Dr.	Sacramento	California	95820	(860) 563-7...	aoja.cashm...
C000010004	Nichole	White	9601 Ocean...	Atwater	California	95301	(212) 555-9...	nichole.whit...
C000010005	Brad	Pitt	853 Stonybr...	Torrance	California	90505	(917) 444-2...	brad.pitt@g...
C000010006	Simon	Britney	69 Cypress St.	Oakland	California	94603	(646) 756-3...	simon.britne...
C000010007	Mia	Shaw	12 Ridgevie...	Fullerton	California	92831	(212) 454-0...	mia.shaw@...
C000010008	Jenn	Diago	9982 Manor ...	San Lorenzo	California	90274	(917) 582-7...	jenn.diago...
C000010009	Gretchen	Sandlin	369 Vemon Dr.	Encino	California	91316	(631) 378-7...	gretchen.sa...
C000010010	Rajat	Namdev	7738 Pheas...	Pearland	Texas	77584	(607) 334-9...	rajat.namde...
C000010011	Dheeraj	Panchal	42 8th St.	Denton	Texas	76205	(845) 316-2...	dheeraj.pan...
C000010012	Kratika	Sharma	350 S. Look...	Houston	Texas	77064	(212) 558-3...	kratika.shar...
C000010013	Kapil	Patidar	46 Oxford S...	Houston	Texas	77092	(917) 867-0...	kapil.patidar...
C000010014	Himanshu	Parashar	451 Temple ...	Mesquite	Texas	75150	(617) 876-3...	himanshu.p...

Step 3: Select the import method - do you want to directly insert or do you want to run the insert script manually and then enter the data. It is required to run manually if there is any change in data that needs to be done. Once, chosen, click Next.


Data Import Wizard - Step 2 of 4

Import Method

Data Preview
Import Method
Column Definition
Finish

Specify the method for importing data. For Staging External Table method, an external table will be created as a staging table for importing the Target Table. For other import methods, data is imported directly into the table.

Import Method: Insert

Send Create Script to SQL Worksheet

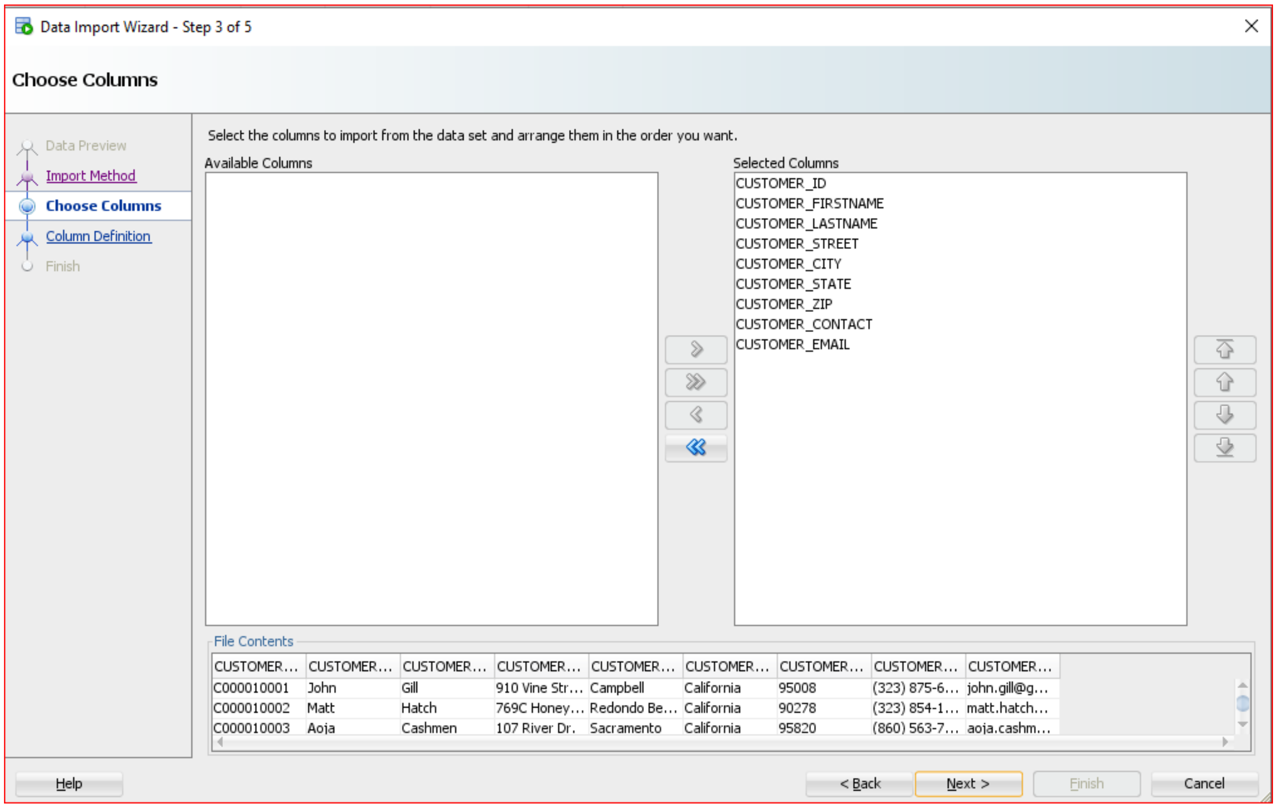
Table Name: CUSTOMER13

Import Row Limit: 100

File Contents

CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...
C000010001	John	Gill	910 Vine Str...	Campbell	California	95008	(323) 875-6...	john.gill@g...
C000010002	Matt	Hatch	769C Honey...	Redondo Be...	California	90278	(323) 854-1...	matt.hatch...
C000010003	Aoja	Cashmen	107 River Dr.	Sacramento	California	95820	(860) 563-7...	aoja.cashm...
C000010004	Nichole	White	9601 Ocean...	Atwater	California	95301	(212) 555-9...	nichole.whit...
C000010005	Brad	Pitt	853 Stonybr...	Torrance	California	90505	(917) 444-2...	brad.pitt@g...
C000010006	Simon	Britney	69 Cypress St.	Oakland	California	94603	(646) 756-3...	simon.britne...
C000010007	Mia	Shaw	12 Ridgevie...	Fullerton	California	92831	(212) 454-0...	mia.shaw@...
C000010008	Jenn	Diago	9982 Manor ...	San Lorenzo	California	90274	(917) 582-7...	jenn.diago...
C000010009	Gretchen	Sandlin	369 Vemon Dr.	Encino	California	91316	(631) 378-7...	gretchen.sa...
C000010010	Rajat	Namdev	7738 Pheas...	Pearland	Texas	77584	(607) 334-9...	rajat.namde...
C000010011	Dheeraj	Panchal	42 8th St.	Denton	Texas	76205	(845) 316-2...	dheeraj.pan...
C000010012	Kratika	Sharma	350 S. Look...	Houston	Texas	77064	(212) 558-3...	kratika.shar...
C000010013	Kapil	Patidar	46 Oxford S...	Houston	Texas	77092	(917) 867-0...	kapil.patidar...
C000010014	Himanshu	Parashar	451 Temple ...	Mesquite	Texas	75150	(617) 876-3...	himanshu.p...
C000010015	Kalyani	Pathak	7068 Elm Rd.	Arlington	Texas	76014	(254) 398-8...	kalyani.path...
C000010016	Aashta	Naik	4983 Clair S...	Waco	Texas	76706	(254) 652-9...	aashta.naik...
C000010017	Minal	Shah	7023 Maple ...	Arlington	Texas	76011	(860) 678-1...	minal.shah...

Step 4: Choose the columns as required. Click Next.


Data Import Wizard - Step 3 of 5

Choose Columns

Data Preview
Import Method
Choose Columns
Column Definition
Finish

Select the columns to import from the data set and arrange them in the order you want.

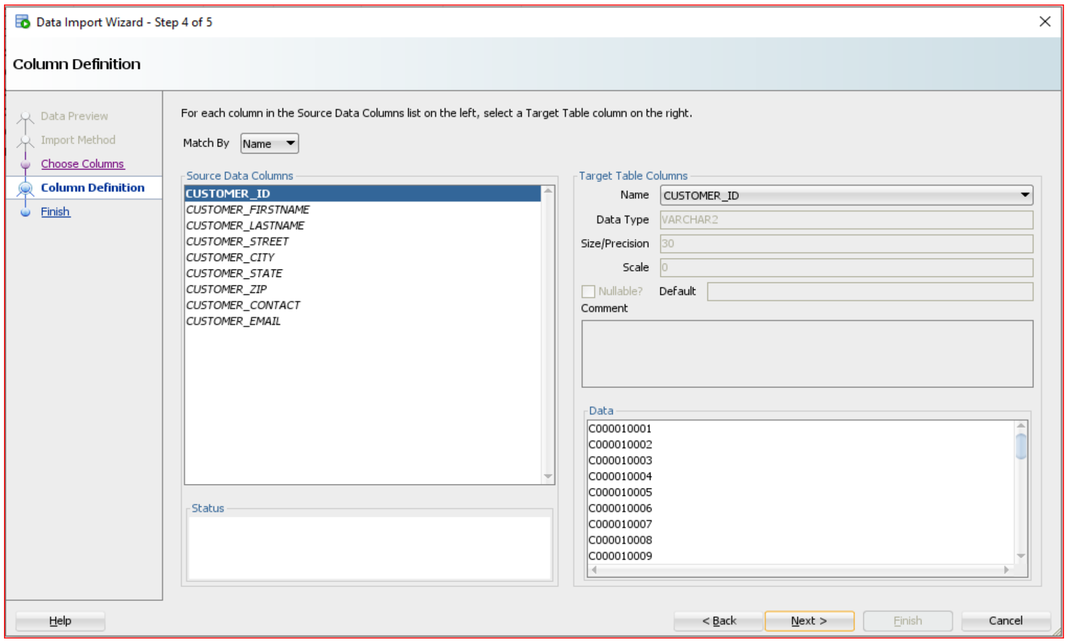
Available Columns

Selected Columns
CUSTOMER_ID
CUSTOMER_FIRSTNAME
CUSTOMER_LASTNAME
CUSTOMER_STREET
CUSTOMER_CITY
CUSTOMER_STATE
CUSTOMER_ZIP
CUSTOMER_CONTACT
CUSTOMER_EMAIL

File Contents

CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...	CUSTOMER...
C000010001	John	Gill	910 Vine Str...	Campbell	California	95008	(323) 875-6...	john.gill@g...
C000010002	Matt	Hatch	769C Honey...	Redondo Be...	California	90278	(323) 854-1...	matt.hatch...
C000010003	Aoja	Cashmen	107 River Dr.	Sacramento	California	95820	(860) 563-7...	aoja.cashm...

Step 5: In this tab, make sure there are no warning signs besides the column names under Source Data Column. It should be exactly and properly mapped to Target table columns as given on right. Proceed once all the warnings are cleared.



Data Import Wizard - Step 4 of 5

Column Definition

For each column in the Source Data Columns list on the left, select a Target Table column on the right.

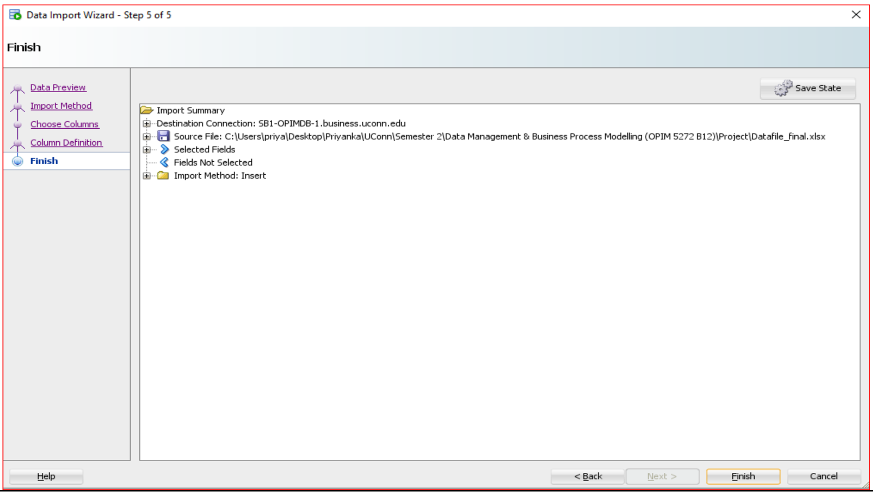
Match By: Name

Source Data Columns:
CUSTOMER_ID
CUSTOMER_FIRSTNAME
CUSTOMER_LASTNAME
CUSTOMER_STREET
CUSTOMER_CITY
CUSTOMER_STATE
CUSTOMER_ZIP
CUSTOMER_CONTACT
CUSTOMER_EMAIL

Target Table Columns
Name: CUSTOMER_ID
Data Type: VARCHAR2
Size/Precision: 30
Scale: 0
Nullable? Default
Comment

Data
C000010001
C000010002
C000010003
C000010004
C000010005
C000010006
C000010007
C000010008
C000010009

Step 6: Once you click “Finish”, the data would be imported.



Data Import Wizard - Step 5 of 5

Finish

Import Summary
Destination Connection: SB1-OPIMDB-1.business.uconn.edu
Source File: C:\Users\priya\Desktop\Priyanka\UConn\Semester 2\Data Management & Business Process Modelling (OPIM 5272 B12)\Project\Datafile_final.xlsx
Selected Fields
Fields Not Selected
Import Method: Insert

We followed these steps to load and import data in all the 15 tables.

Reports

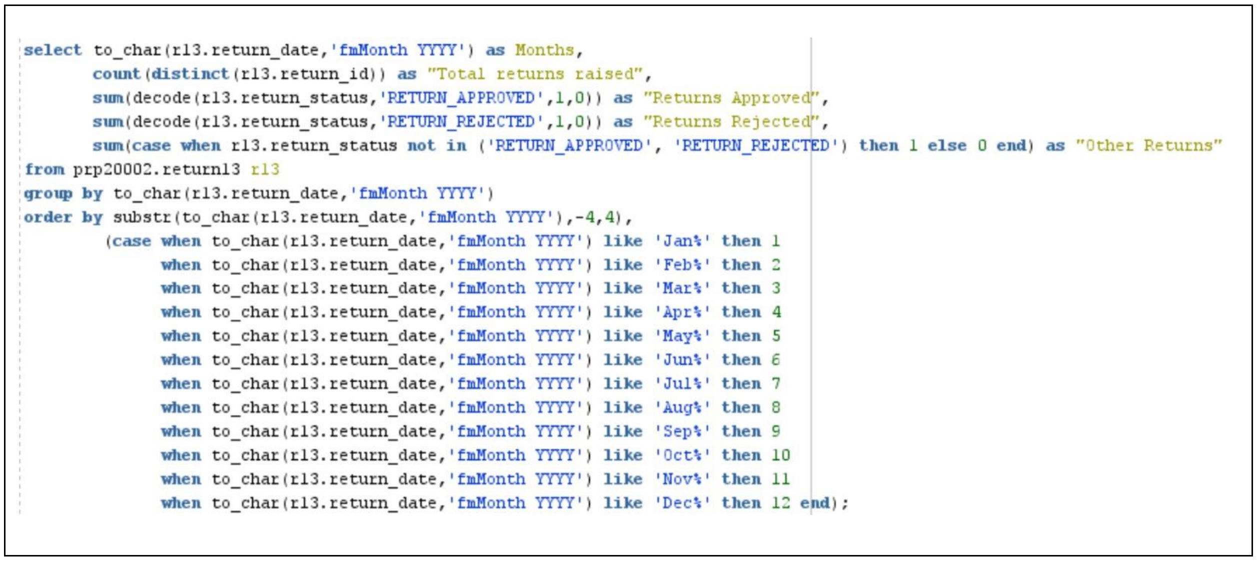
2.1 Report 1

- **Purpose:** To analyze and compare month wise count of number of returns raised, total number of returns approved, and total number of returns rejected.

- **Benefit:** Helps track historical and current business performance. Also helps in predicting month end numbers and serves as data for planning into the future.

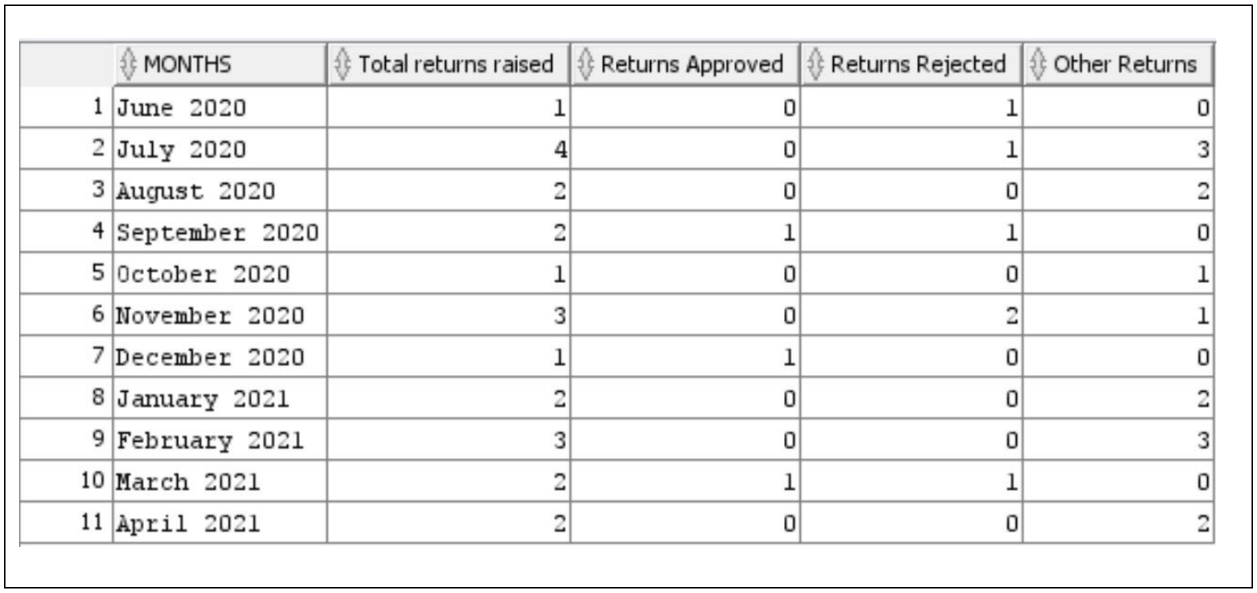
- **Use in business metric:** This report enables us to see as to how many total returns were created in a month, thereby enabling us to see the pattern of return creation and compare the same month wise. Apart from this, it further shows us as to how many returns from those created, were approved and rejected. Helps in tracking business performance. Can help in pointing out problems within the return process. Returns approved, rejected, and those not terminated gives a view of overall return performance. It also showcases where the bottleneck in the system might be. Too many rejected returns can lead to poor customer overview, whereas too less rejected returns can mean angry sellers and a poor profit/loss performance.

- **SQL Query:**



```
select to_char(r13.return_date,'fmMonth YYYY') as Months,
       count(distinct(r13.return_id)) as "Total returns raised",
       sum(decode(r13.return_status,'RETURN_APPROVED',1,0)) as "Returns Approved",
       sum(decode(r13.return_status,'RETURN_REJECTED',1,0)) as "Returns Rejected",
       sum(case when r13.return_status not in ('RETURN_APPROVED', 'RETURN_REJECTED') then 1 else 0 end) as "Other Returns"
from prp20002.return13 r13
group by to_char(r13.return_date,'fmMonth YYYY')
order by substr(to_char(r13.return_date,'fmMonth YYYY'),-4,4),
        (case when to_char(r13.return_date,'fmMonth YYYY') like 'Jan%' then 1
              when to_char(r13.return_date,'fmMonth YYYY') like 'Feb%' then 2
              when to_char(r13.return_date,'fmMonth YYYY') like 'Mar%' then 3
              when to_char(r13.return_date,'fmMonth YYYY') like 'Apr%' then 4
              when to_char(r13.return_date,'fmMonth YYYY') like 'May%' then 5
              when to_char(r13.return_date,'fmMonth YYYY') like 'Jun%' then 6
              when to_char(r13.return_date,'fmMonth YYYY') like 'Jul%' then 7
              when to_char(r13.return_date,'fmMonth YYYY') like 'Aug%' then 8
              when to_char(r13.return_date,'fmMonth YYYY') like 'Sep%' then 9
              when to_char(r13.return_date,'fmMonth YYYY') like 'Oct%' then 10
              when to_char(r13.return_date,'fmMonth YYYY') like 'Nov%' then 11
              when to_char(r13.return_date,'fmMonth YYYY') like 'Dec%' then 12 end);
```

- **Output:**

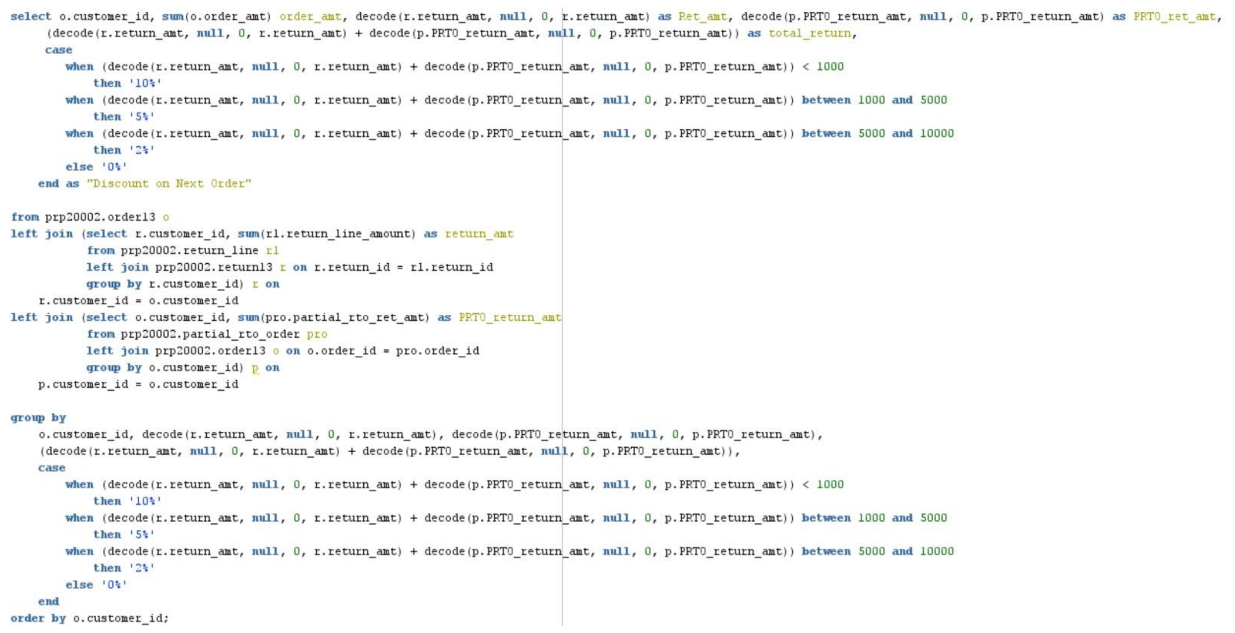


	MONTHS	Total returns raised	Returns Approved	Returns Rejected	Other Returns
1	June 2020	1	0	1	0
2	July 2020	4	0	1	3
3	August 2020	2	0	0	2
4	September 2020	2	1	1	0
5	October 2020	1	0	0	1
6	November 2020	3	0	2	1
7	December 2020	1	1	0	0
8	January 2021	2	0	0	2
9	February 2021	3	0	0	3
10	March 2021	2	1	1	0
11	April 2021	2	0	0	2

2.2 Report 2

- **Purpose:** Incentivize customers for better return performance
- **Benefit:** Customer retention and promoting better buyers to spend more on platform
- **Use in business metric**: Customers with less returns leads to a better profit and loss statement for the company. In lieu of this, the company is incentivizing customers with better behavior by offering them more discounts and ensuring they retain the business with them. This analysis can also be used to show customers how far they are from a higher tier and to educate them on better return practices.

- **SQL Query:**

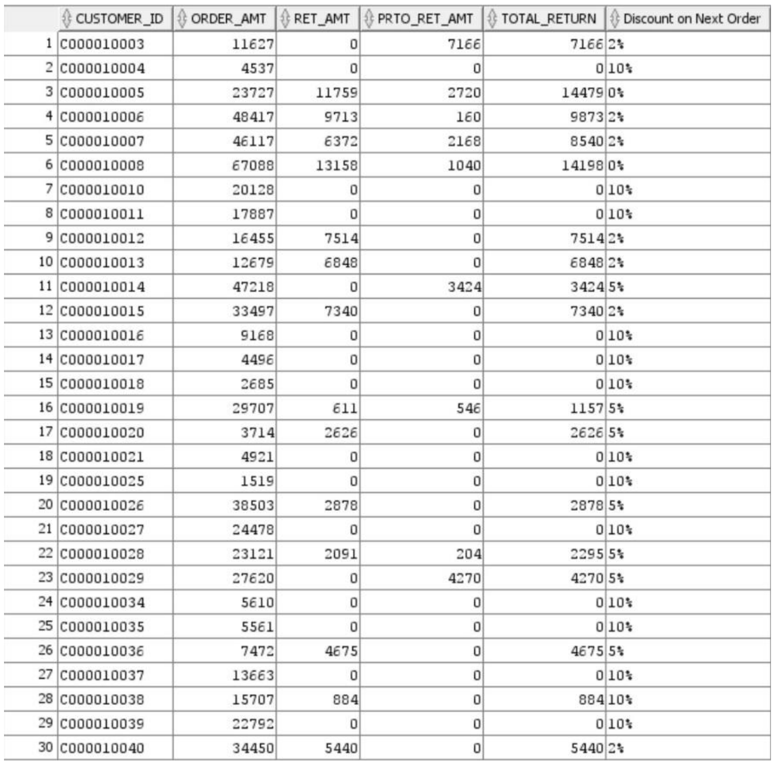


```sql
select o.customer_id, sum(o.order_amt) order_amt, decode(r.return_amt, null, 0, r.return_amt) as Ret_amt, decode(p.PRT0_return_amt, null, 0, p.PRT0_return_amt) as PRT0_ret_amt,
     (decode(r.return_amt, null, 0, r.return_amt) + decode(p.PRT0_return_amt, null, 0, p.PRT0_return_amt)) as total_return,
     case
        when (decode(r.return_amt, null, 0, r.return_amt) + decode(p.PRT0_return_amt, null, 0, p.PRT0_return_amt)) < 1000
            then '10%'
        when (decode(r.return_amt, null, 0, r.return_amt) + decode(p.PRT0_return_amt, null, 0, p.PRT0_return_amt)) between 1000 and 5000
            then '5%'
        when (decode(r.return_amt, null, 0, r.return_amt) + decode(p.PRT0_return_amt, null, 0, p.PRT0_return_amt)) between 5000 and 10000
            then '2%'
        else '0%'
    end as "Discount on Next Order"

from prp20002.order13 o
left join (select r.customer_id, sum(rl.return_line_amount) as return_amt
           from prp20002.return_line rl
           left join prp20002.return13 r on r.return_id = rl.return_id
           group by r.customer_id) r on
    r.customer_id = o.customer_id
left join (select o.customer_id, sum(pro.partial_rto_ret_amt) as PRT0_return_amt
           from prp20002.partial_rto_order pro
           left join prp20002.order13 o on o.order_id = pro.order_id
           group by o.customer_id) p on
    p.customer_id = o.customer_id

group by
    o.customer_id, decode(r.return_amt, null, 0, r.return_amt), decode(p.PRT0_return_amt, null, 0, p.PRT0_return_amt),
    (decode(r.return_amt, null, 0, r.return_amt) + decode(p.PRT0_return_amt, null, 0, p.PRT0_return_amt)),
    case
        when (decode(r.return_amt, null, 0, r.return_amt) + decode(p.PRT0_return_amt, null, 0, p.PRT0_return_amt)) < 1000
            then '10%'
        when (decode(r.return_amt, null, 0, r.return_amt) + decode(p.PRT0_return_amt, null, 0, p.PRT0_return_amt)) between 1000 and 5000
            then '5%'
        when (decode(r.return_amt, null, 0, r.return_amt) + decode(p.PRT0_return_amt, null, 0, p.PRT0_return_amt)) between 5000 and 10000
            then '2%'
        else '0%'
    end
order by o.customer_id;
```

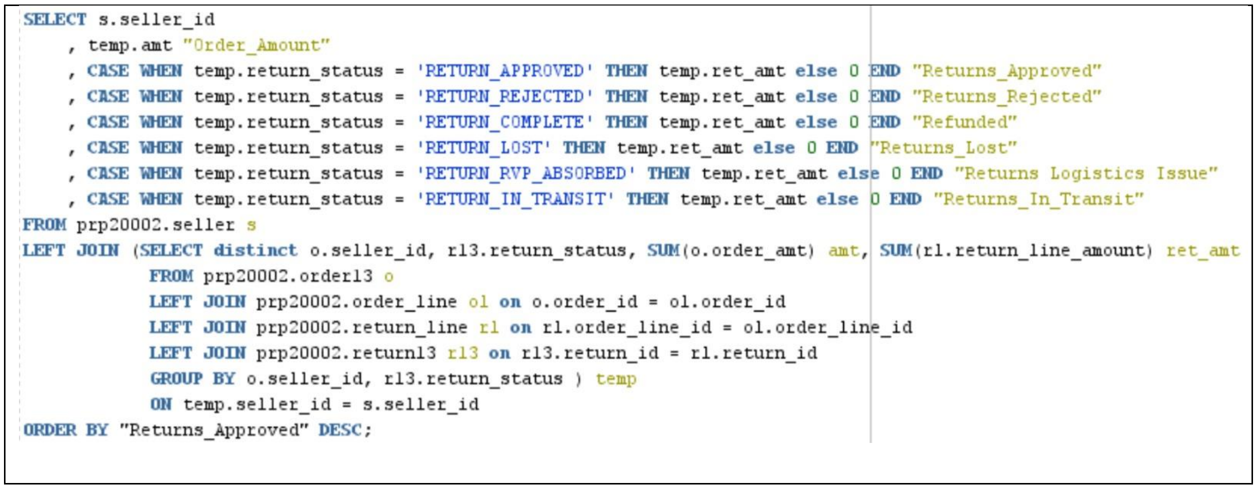
- **Output:**



	CUSTOMER_ID	ORDER_AMT	RET_AMT	PRTO_RET_AMT	TOTAL_RETURN	Discount on Next Order
1	C000010003	11627	0	7166	7166	2%
2	C000010004	4537	0	0	0	10%
3	C000010005	23727	11759	2720	14479	0%
4	C000010006	48417	9713	160	9873	2%
5	C000010007	46117	6372	2168	8540	2%
6	C000010008	67088	13158	1040	14198	0%
7	C000010010	20128	0	0	0	10%
8	C000010011	17887	0	0	0	10%
9	C000010012	16455	7514	0	7514	2%
10	C000010013	12679	6848	0	6848	2%
11	C000010014	47218	0	3424	3424	5%
12	C000010015	33497	7340	0	7340	2%
13	C000010016	9168	0	0	0	10%
14	C000010017	4496	0	0	0	10%
15	C000010018	2685	0	0	0	10%
16	C000010019	29707	611	546	1157	5%
17	C000010020	3714	2626	0	2626	5%
18	C000010021	4921	0	0	0	10%
19	C000010025	1519	0	0	0	10%
20	C000010026	38503	2878	0	2878	5%
21	C000010027	24478	0	0	0	10%
22	C000010028	23121	2091	204	2295	5%
23	C000010029	27620	0	4270	4270	5%
24	C000010034	5610	0	0	0	10%
25	C000010035	5561	0	0	0	10%
26	C000010036	7472	4675	0	4675	5%
27	C000010037	13663	0	0	0	10%
28	C000010038	15707	884	0	884	10%
29	C000010039	22792	0	0	0	10%
30	C000010040	34450	5440	0	5440	2%

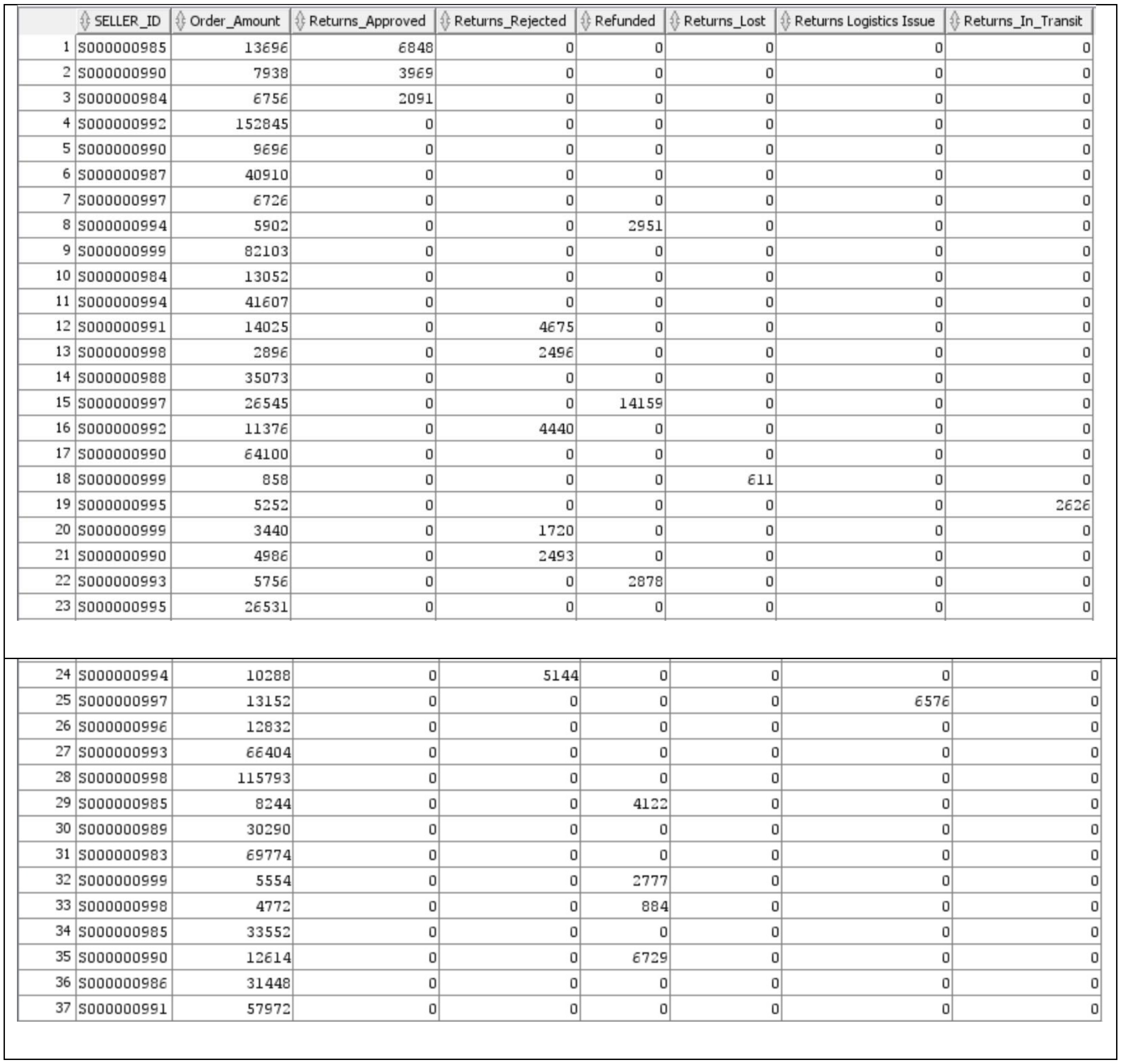
2.3 Report 3

- **Purpose:** Seller dashboard
- **Benefit:** Can be used to monitor seller performance
- **Use in business metric**: This can be used to track and measure seller's returns performance. A seller with high return can be detrimental to the business. Catching such a seller behavior early is beneficial to the business. This can also be used to incentivize sellers in the future and decide if we need to part ways with a seller or not.
- **SQL Query:**



```
SELECT s.seller_id
    , temp.amt "Order_Amount"
    , CASE WHEN temp.return_status = 'RETURN_APPROVED' THEN temp.ret_amt else 0 END "Returns_Approved"
    , CASE WHEN temp.return_status = 'RETURN_REJECTED' THEN temp.ret_amt else 0 END "Returns_Rejected"
    , CASE WHEN temp.return_status = 'RETURN_COMPLETE' THEN temp.ret_amt else 0 END "Refunded"
    , CASE WHEN temp.return_status = 'RETURN_LOST' THEN temp.ret_amt else 0 END "Returns_Lost"
    , CASE WHEN temp.return_status = 'RETURN_RVP_ABSORBED' THEN temp.ret_amt else 0 END "Returns Logistics Issue"
    , CASE WHEN temp.return_status = 'RETURN_IN_TRANSIT' THEN temp.ret_amt else 0 END "Returns_In_Transit"
FROM prp20002.seller s
LEFT JOIN (SELECT distinct o.seller_id, r13.return_status, SUM(o.order_amt) amt, SUM(rl.return_line_amount) ret_amt
            FROM prp20002.order13 o
            LEFT JOIN prp20002.order_line ol on o.order_id = ol.order_id
            LEFT JOIN prp20002.return_line rl on rl.order_line_id = ol.order_line_id
            LEFT JOIN prp20002.return13 r13 on r13.return_id = rl.return_id
            GROUP BY o.seller_id, r13.return_status ) temp
            ON temp.seller_id = s.seller_id
ORDER BY "Returns_Approved" DESC;
```

- **Output:**

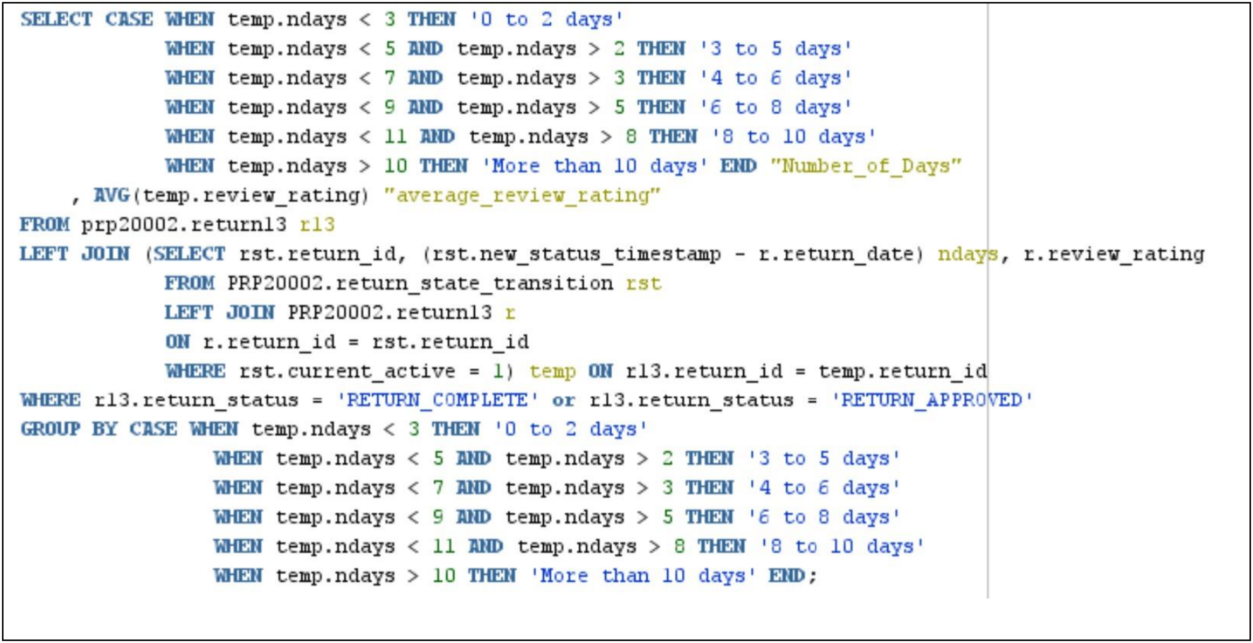


	SELLER_ID	Order_Amount	Returns_Approved	Returns_Rejected	Refunded	Returns_Lost	Returns Logistics Issue	Returns_In_Transit
1	S000000985	13696	6848	0	0	0	0	0
2	S000000990	7938	3969	0	0	0	0	0
3	S000000984	6756	2091	0	0	0	0	0
4	S000000992	152845	0	0	0	0	0	0
5	S000000990	9696	0	0	0	0	0	0
6	S000000987	40910	0	0	0	0	0	0
7	S000000997	6726	0	0	0	0	0	0
8	S000000994	5902	0	0	2951	0	0	0
9	S000000999	82103	0	0	0	0	0	0
10	S000000984	13052	0	0	0	0	0	0
11	S000000994	41607	0	0	0	0	0	0
12	S000000991	14025	0	4675	0	0	0	0
13	S000000998	2896	0	2496	0	0	0	0
14	S000000988	35073	0	0	0	0	0	0
15	S000000997	26545	0	0	14159	0	0	0
16	S000000992	11376	0	4440	0	0	0	0
17	S000000990	64100	0	0	0	0	0	0
18	S000000999	858	0	0	0	611	0	0
19	S000000995	5252	0	0	0	0	0	2626
20	S000000999	3440	0	1720	0	0	0	0
21	S000000990	4986	0	2493	0	0	0	0
22	S000000993	5756	0	0	2878	0	0	0
23	S000000995	26531	0	0	0	0	0	0
24	S000000994	10288	0	5144	0	0	0	0
25	S000000997	13152	0	0	0	0	6576	0
26	S000000996	12832	0	0	0	0	0	0
27	S000000993	66404	0	0	0	0	0	0
28	S000000998	115793	0	0	0	0	0	0
29	S000000985	8244	0	0	4122	0	0	0
30	S000000989	30290	0	0	0	0	0	0
31	S000000983	69774	0	0	0	0	0	0
32	S000000999	5554	0	0	2777	0	0	0
33	S000000998	4772	0	0	884	0	0	0
34	S000000985	33552	0	0	0	0	0	0
35	S000000990	12614	0	0	6729	0	0	0
36	S000000986	31448	0	0	0	0	0	0
37	S000000991	57972	0	0	0	0	0	0

2.4 Report 4

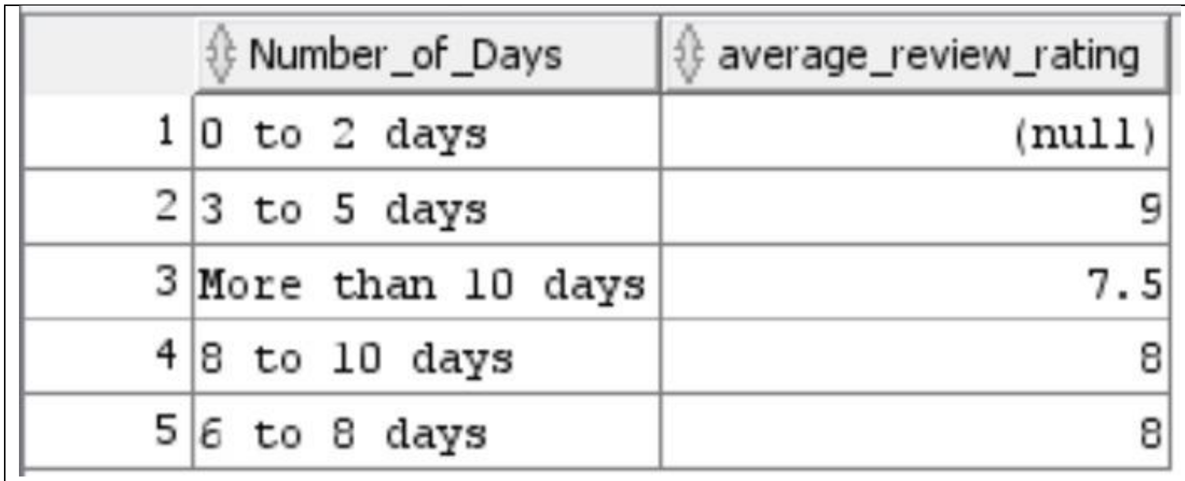
- **Purpose:** Customer perception of return as time taken to complete increases
- **Benefit:** Derive insights to improve business performance
- **Use in business metric**: Gives an aim for the timeline to target to complete a return to increase customer satisfaction. A better return experience increases customer engagement and trust with the platform and promotes healthy return behavior.

- **SQL Query:**



```
SELECT CASE WHEN temp.ndays < 3 THEN '0 to 2 days'
            WHEN temp.ndays < 5 AND temp.ndays > 2 THEN '3 to 5 days'
            WHEN temp.ndays < 7 AND temp.ndays > 3 THEN '4 to 6 days'
            WHEN temp.ndays < 9 AND temp.ndays > 5 THEN '6 to 8 days'
            WHEN temp.ndays < 11 AND temp.ndays > 8 THEN '8 to 10 days'
            WHEN temp.ndays > 10 THEN 'More than 10 days' END "Number_of_Days"
    , AVG(temp.review_rating) "average_review_rating"
FROM prp20002.return13 r13
LEFT JOIN (SELECT rst.return_id, (rst.new_status_timestamp - r.return_date) ndays, r.review_rating
            FROM PRP20002.return_state_transition rst
            LEFT JOIN PRP20002.return13 r
            ON r.return_id = rst.return_id
            WHERE rst.current_active = 1) temp ON r13.return_id = temp.return_id
WHERE r13.return_status = 'RETURN_COMPLETE' or r13.return_status = 'RETURN_APPROVED'
GROUP BY CASE WHEN temp.ndays < 3 THEN '0 to 2 days'
                WHEN temp.ndays < 5 AND temp.ndays > 2 THEN '3 to 5 days'
                WHEN temp.ndays < 7 AND temp.ndays > 3 THEN '4 to 6 days'
                WHEN temp.ndays < 9 AND temp.ndays > 5 THEN '6 to 8 days'
                WHEN temp.ndays < 11 AND temp.ndays > 8 THEN '8 to 10 days'
                WHEN temp.ndays > 10 THEN 'More than 10 days' END;
```

- **Output:**



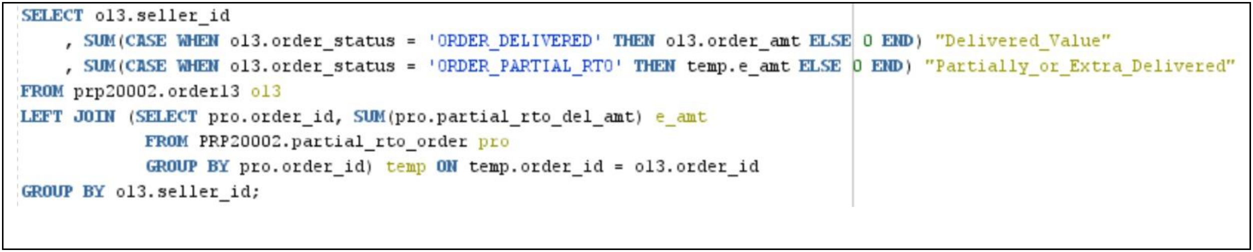
	Number_of_Days	average_review_rating
1	0 to 2 days	(null)
2	3 to 5 days	9
3	More than 10 days	7.5
4	8 to 10 days	8
5	6 to 8 days	8

2.5 Report 5

- **Purpose:** Test performance for the added feature
- **Benefit:** Partial RTO helps to avoid additional or repetitive logistics cost and helps us deliver more when instead a return would have happened

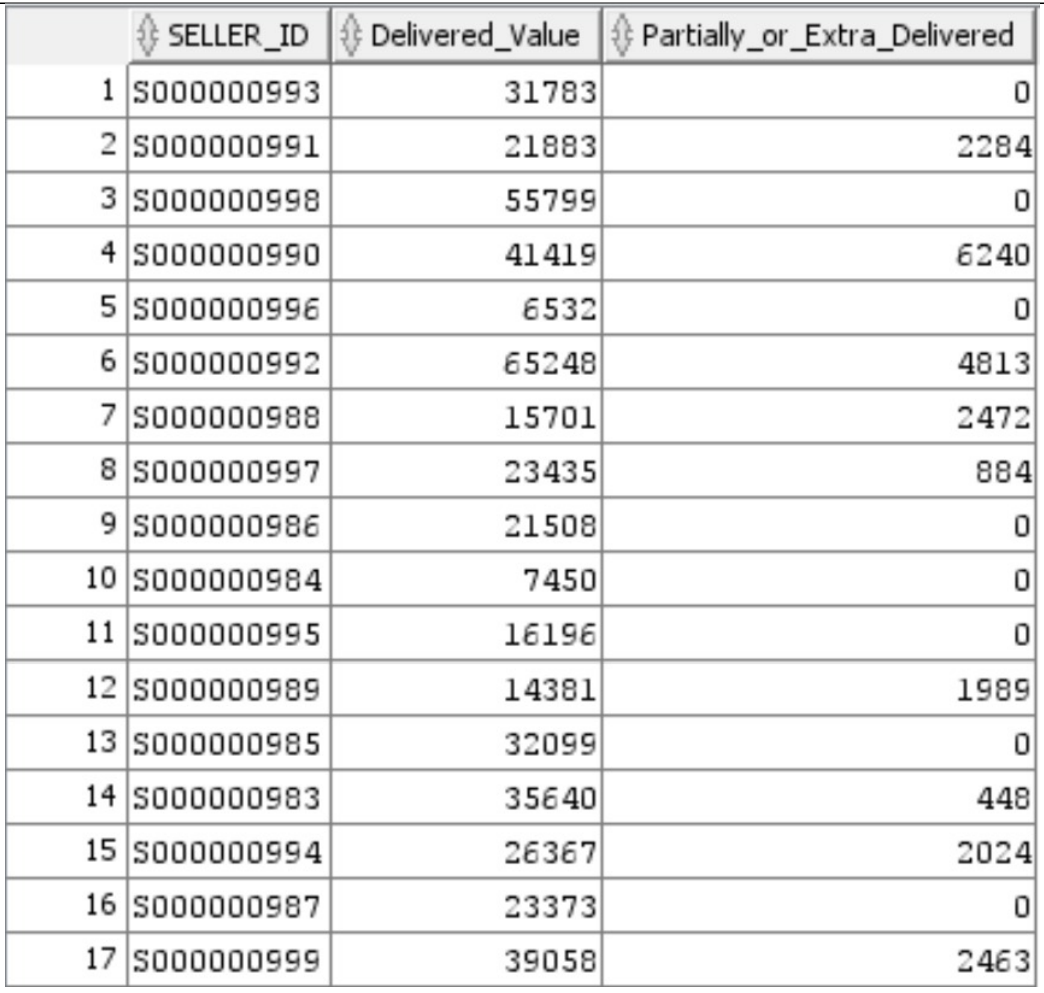
- **Use in business metric**: In case the buyer doesn't want the complete order, and the cost for taking the order back to warehouse is significant due to B2B market, a partial invoice is generated and the customer can keep the part of the order they like. This helps deliver more product and make sure of the already invested logistics cost for that particular order.

- **SQL Query:**



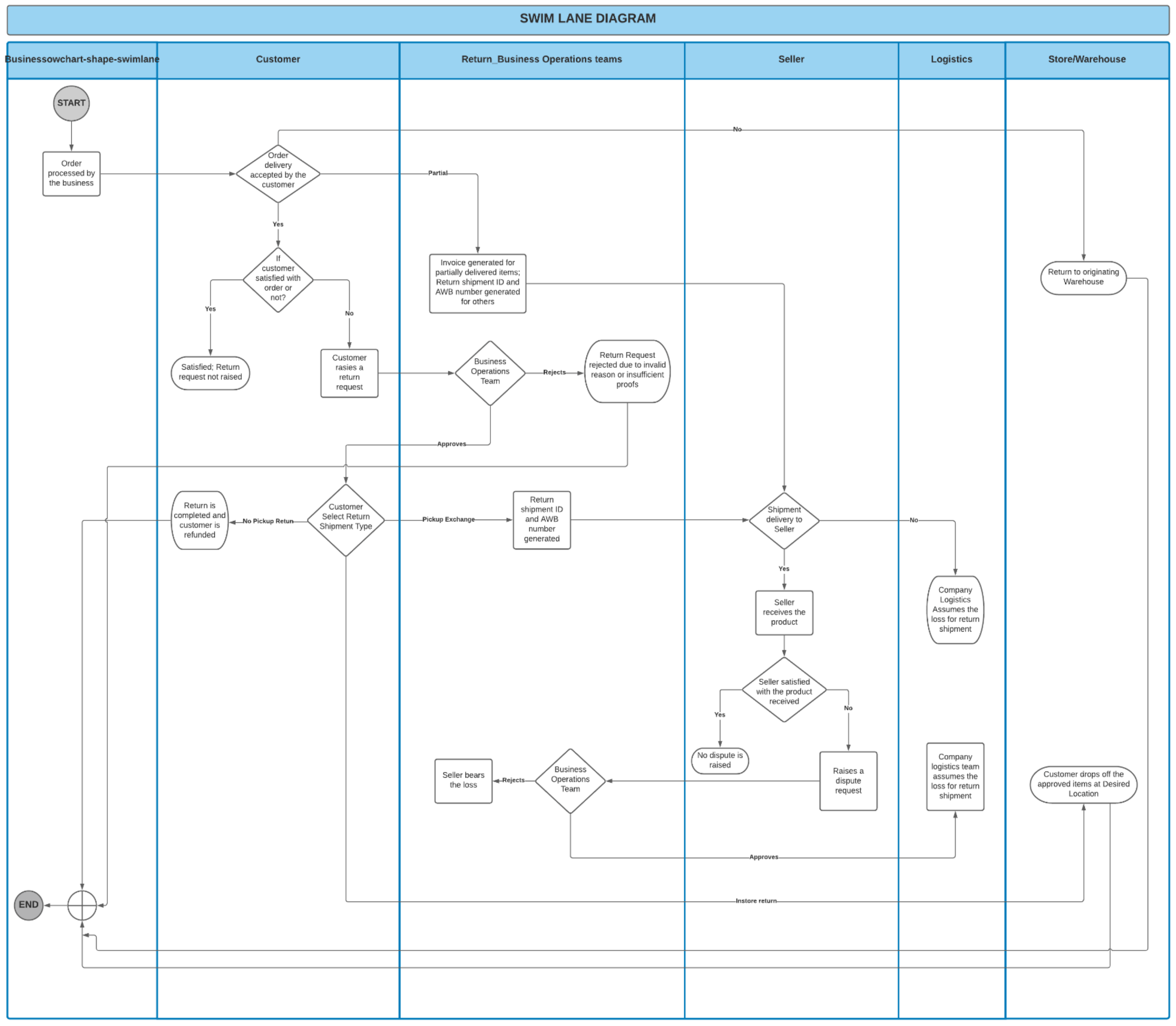
```
SELECT o13.seller_id
    , SUM(CASE WHEN o13.order_status = 'ORDER_DELIVERED' THEN o13.order_amt ELSE 0 END) "Delivered_Value"
    , SUM(CASE WHEN o13.order_status = 'ORDER_PARTIAL_RTO' THEN temp.e_amt ELSE 0 END) "Partially_or_Extra_Delivered"
FROM prp20002.order13 o13
LEFT JOIN (SELECT pro.order_id, SUM(pro.partial_rto_del_amt) e_amt
           FROM PRP20002.partial_rto_order pro
           GROUP BY pro.order_id) temp ON temp.order_id = o13.order_id
GROUP BY o13.seller_id;
```

- **Output:**



	SELLER_ID	Delivered_Value	Partially_or_Extra_Delivered
1	S000000993	31783	0
2	S000000991	21883	2284
3	S000000998	55799	0
4	S000000990	41419	6240
5	S000000996	6532	0
6	S000000992	65248	4813
7	S000000988	15701	2472
8	S000000997	23435	884
9	S000000986	21508	0
10	S000000984	7450	0
11	S000000995	16196	0
12	S000000989	14381	1989
13	S000000985	32099	0
14	S000000983	35640	448
15	S000000994	26367	2024
16	S000000987	23373	0
17	S000000999	39058	2463

Swim Lane Diagram



SWIM LANE DIAGRAM
Businessowchart-shape-swimlane
Customer
Return_Business Operations teams
Seller
Logistics
Store/Warehouse
START
Order processed by the business
Order delivery accepted by the customer
No
Partial
Yes
If customer satisfied with order or not?
Invoice generated for partially delivered items; Return shipment ID and AWB number generated for others
Return to originating Warehouse
Yes
No
Satisfied; Return request not raised
Customer rasies a return request
Business Operations Team
Rejects
Return Request rejected due to invalid reason or insufficient proofs
Approves
Return is completed and customer is refunded
No Pickup Retun
Customer Select Return Shipment Type
Pickup Exchange
Return shipment ID and AWB number generated
Shipment delivery to Seller
No
Yes
Seller receives the product
Company Logistics Assumes the loss for return shipment
Seller satisfied with the product received
Yes
No
No dispute is raised
Seller bears the loss
Rejects
Business Operations Team
Raises a dispute request
Company logistics team assumes the loss for return shipment
Customer drops off the approved items at Desired Location
Approves
END
Instore return