YASHASVI THANAI
MS in Business Analytics

HW3 due by Wednesday Oct. 12 by 11:59 PM

The create_expense.sql script (you can find it on Blackboard in the Homework 3 folder) creates a database which contains the 7 tables described below. The data contains a tracking system for expense reports filed by employees at a manufacturing company. **Please watch the Panopto video for week 5 before doing this Homework 3.**

employees	**Field Description**
Ssn (pk)	Unique SSN ID# for employee
First_name	Employee first name
Last_name	Employee last_name
Dept (fk)	Dept ID#
Start_year	Year of employment

trips	**Field Description**
Employee (pk, fk)	SSN of employee travelling
Trip_ID (pk)	Unique Trip ID#
Start_date	Start date of trip
End_date	End date of trip
Reason_code (fk)	Code for reason for trip

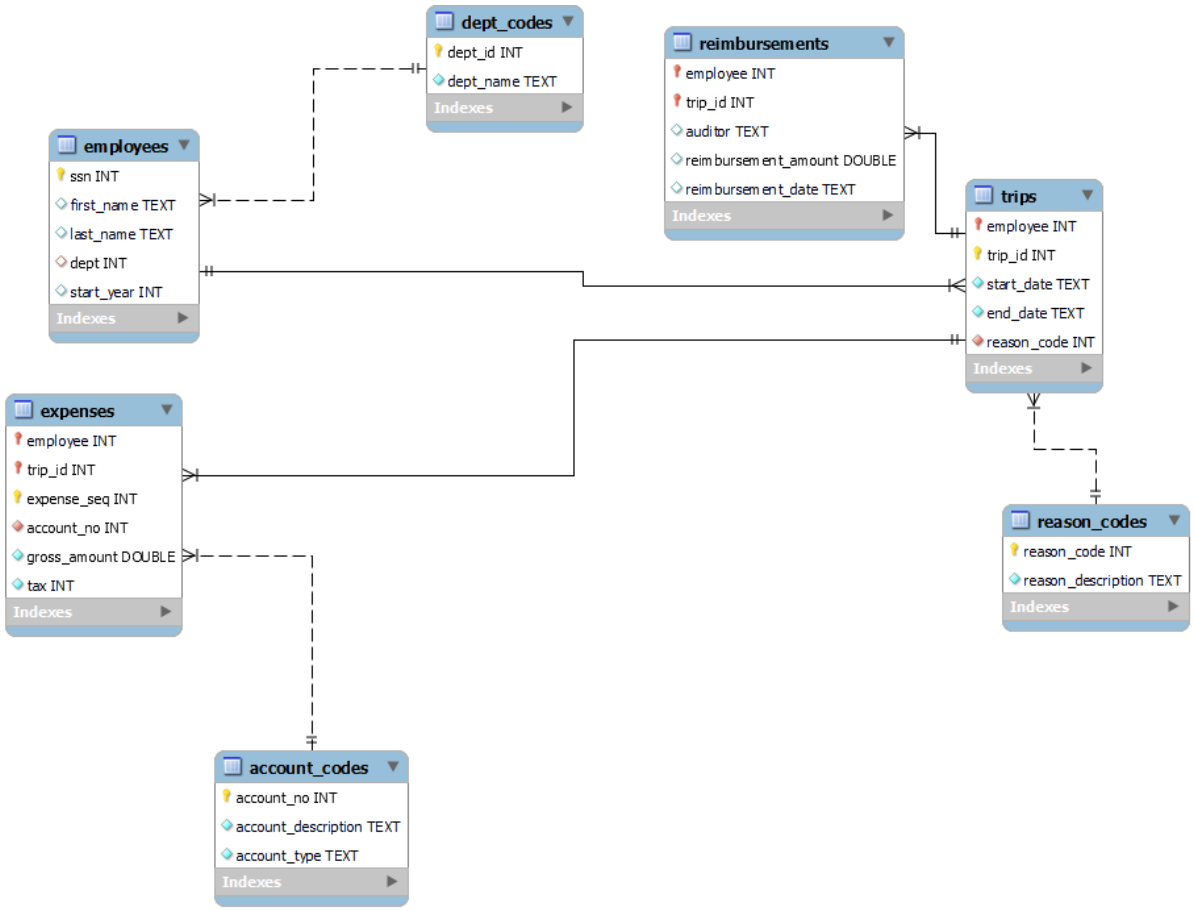
expenses	**Field Description**
Employee (pk, fk)	SSN of employee travelling
Trip_id (pk, fk)	Unique Trip ID#
Expense_seq (pk)	Sequence# for expense report line item
Account_no (fk)	Account number for line item
Gross_amount	Gross dollar amount of line item
tax	Sales tax (if applicable) of line item

dept_codes	**Field Description**
Dept_ID (pk)	Dept ID#
Dept_name	Name of department

Reason_codes	**Field Description**
Reason_code (pk)	Reason ID#
Reason_description	Description of reason for trip

account_codes	**Field Description**
Account_no (pk)	Account ID#
Account_description	Description of account
Account_type	Category of account

reimbursements	**Field Description**
Employee (pk, fk)	SSN of employee travelling
Trip_id (pk, fk)	Unique Trip ID#
Auditor	Auditor last name
Reimbursement_amount	Amount of reimbursement
Reimbursement_date	Date of reimbursement



Please put all of your work into **this single Word doc**.

1. (60 points) Design and create a data warehouse for the Expense database. The decisions about which fields to include and how to aggregate the data are left to you. You do not need to include every single data point from the 7 tables given. Use your judgement as to what will be interesting/useful for the organization. But please make sure that you pull (combine) data from **at least four tables** and compute relevant aggregate statistics. Please see many examples from class lectures and you may adapt those codes for your purpose (for this dataset).

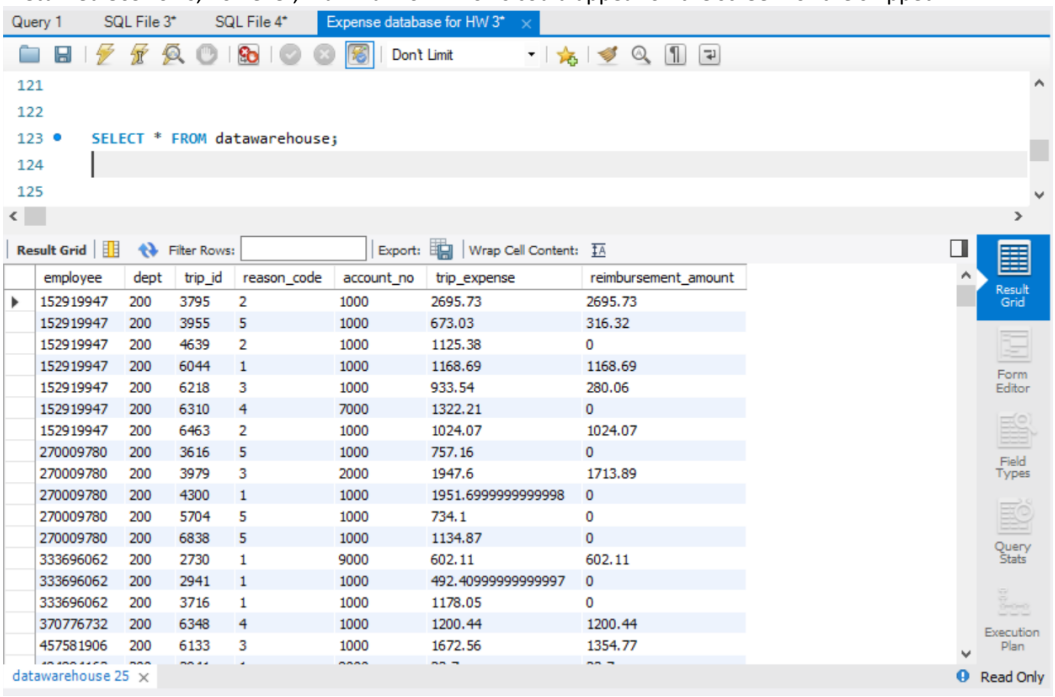
Submit a screenshot of the first 25 rows of your data warehouse (paste into this Word document) and the SQL code that you used to create it. Please copy and paste your SQL code into this Word document.

ANSWER 1

```
CREATE TABLE datawarehouse (
SELECT expenses.employee, department.dept, expenses.trip_id, trips.reason_code,
expenses.account_no, SUM(expenses.gross_amount) AS trip_expense,
reimbursements.reimbursement_amount
FROM expenses
JOIN trips
ON expenses.trip_id = trips.trip_id AND expenses.employee = trips.employee
JOIN (SELECT ssn, dept FROM employees) AS department
ON department.ssn = expenses.employee
JOIN reimbursements
ON reimbursements.employee = expenses.employee AND reimbursements.trip_id = expenses.trip_id
GROUP BY employee, trip_id
ORDER BY dept, employee, trip_id);

SELECT * FROM datawarehouse;
```

*returned 869 rows, however, maximum of 17 rows could appear on the screen for the snippet.



employee	dept	trip_id	reason_code	account_no	trip_expense	reimbursement_amount
152919947	200	3795	2	1000	2695.73	2695.73
152919947	200	3955	5	1000	673.03	316.32
152919947	200	4639	2	1000	1125.38	0
152919947	200	6044	1	1000	1168.69	1168.69
152919947	200	6218	3	1000	933.54	280.06
152919947	200	6310	4	7000	1322.21	0
152919947	200	6463	2	1000	1024.07	1024.07
270009780	200	3616	5	1000	757.16	0
270009780	200	3979	3	2000	1947.6	1713.89
270009780	200	4300	1	1000	1951.6999999999998	0
270009780	200	5704	5	1000	734.1	0
270009780	200	6838	5	1000	1134.87	0
333696062	200	2730	1	9000	602.11	602.11
333696062	200	2941	1	1000	492.40999999999997	0
333696062	200	3716	1	1000	1178.05	0
370776732	200	6348	4	1000	1200.44	1200.44
457581906	200	6133	3	1000	1672.56	1354.77

The datawarehouse uses four main tables to be created. These tables are expenses, employees, trips and reimbursements. The remaining three tables, which are account_codes, reason_codes and dept_codes can be linked to our datawarehouse using their primary keys, account_no, reason_code and dept_id.

Some general aggregates that can be calculated based on this dataware house:

1. Total expenses and reimbursements by department and employee

```
SELECT dept, employee, SUM(trip_expense) AS total_expense , SUM(reimbursement_amount)
AS total_reimbursement
FROM datawarehouse
GROUP BY dept, employee WITH ROLLUP ;
```

2. Total expenses and reimbursements by department and reason

```
SELECT dept, reason_code, SUM(trip_expense) AS total_expense ,
SUM(reimbursement_amount) AS total_reimbursement
FROM datawarehouse
GROUP BY dept, reason_code WITH ROLLUP ;
```

3. Total trips by department

```
SELECT dept, COUNT(dept) AS total_dept_trips
FROM datawarehouse
GROUP BY dept
ORDER BY total_dept_trips DESC;
```

4. Total number of distinct trips

```
SELECT COUNT(DISTINCT trip_id)
FROM datawarehouse;
```

2. (40 points) Create **four** SQL queries on your data warehouse that answer interesting questions. At least two queries should be more than simple queries. For example, more complex queries could include Joins, a Group By element or a subquery or use some aggregate functions and summary calculations (see examples in the class lectures' slides).

Submit a copy of each query SQL code (paste into this Word document), and the screenshot of each query results (or the first 25 rows if it is longer) and a one or two sentence description of the question your SQL code was addressing and what you found in the results.

ANSWER 2

Query 1: Computing the average expenses of each employee and the average number of days they take on their trips for each reason of the trip.

```
SELECT datawarehouse.employee, datawarehouse.trip_id, datawarehouse.reason_code,
reason_codes.reason_description,  AVG(trip_expense) AS average_expense,
AVG(DATEDIFF(STR_TO_DATE (trips.end_date, "%m/%d/%Y"), STR_TO_DATE(trips.start_date,
"%m/%d/%Y"))) AS average_days
FROM datawarehouse
JOIN trips
ON datawarehouse.employee = trips.employee AND datawarehouse.trip_id = trips.trip_id
LEFT JOIN reason_codes
ON reason_codes.reason_code = datawarehouse.reason_code
GROUP BY employee, reason_code
ORDER BY employee, reason_code;
```

Query 1 | SQL File 3* | SQL File 4* | Expense database for HW 3*

Don't Limit

```
137 SELECT datawarehouse.employee, datawarehouse.trip_id, datawarehouse.reason_code, reason_codes.reason_descr
138 AVG(DATEDIFF(str_to_date(trips.end_date, "%m/%d/%Y"), str_to_date(trips.start_date, "%m/%d/%Y"))) AS avera
139 FROM datawarehouse
140 JOIN trips
141 ON datawarehouse.employee = trips.employee AND datawarehouse.trip_id = trips.trip_id
```

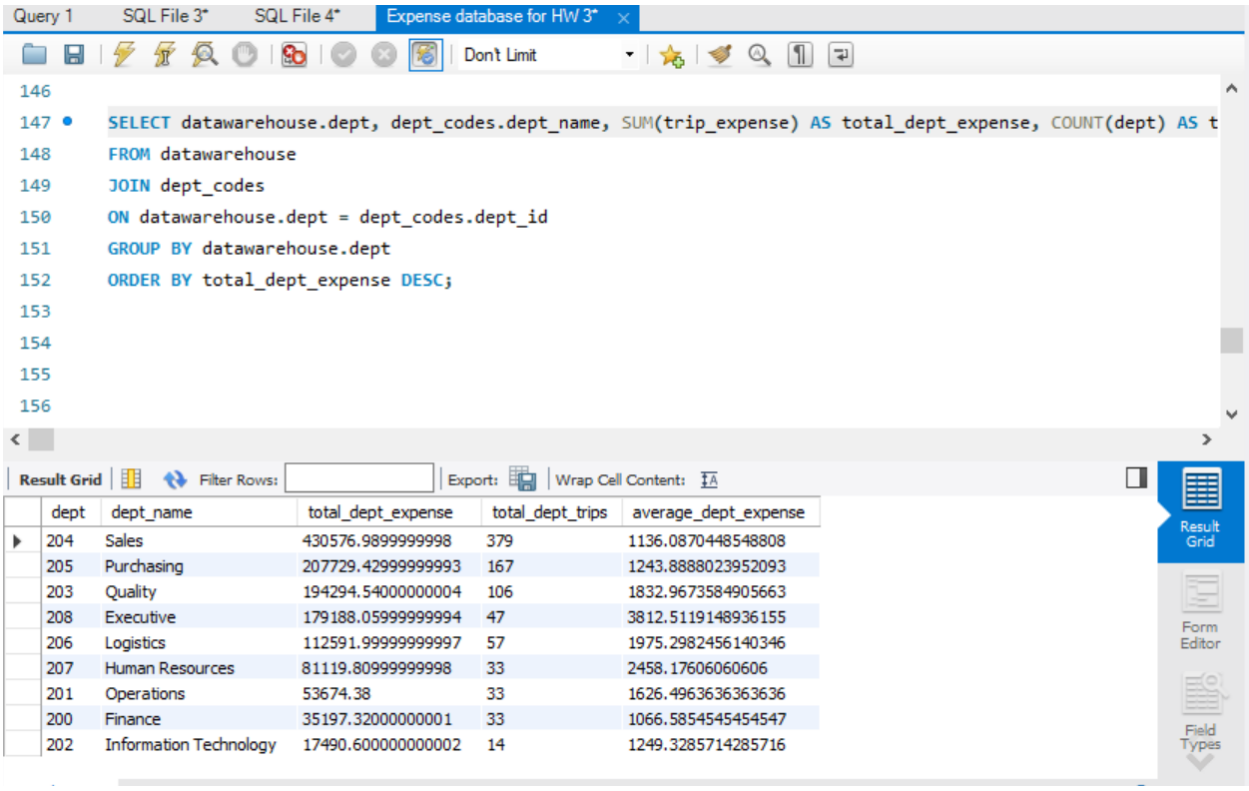
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee	trip_id	reason_code	reason_description	average_expense	average_days
103489296	3256	1	sales call	688.1374999999999	1.5000
103489296	4047	2	conference	3007.38	2.5000
103489296	2968	3	site visit	1042.09	2.0000
103489296	3179	4	off-site meeting	899.1766666666667	2.0000
103489296	4686	5	supplier visit	562.51	2.0000
125144738	7057	2	conference	3177.1800000000003	4.0000
125144738	3803	3	site visit	1412.928	3.2000
125144738	2734	4	off-site meeting	1492.3200000000002	3.0000
125144738	4857	5	supplier visit	1841.23	3.0000
127358183	3686	2	conference	2383.09	2.5000
127358183	3451	3	site visit	756.42	2.0000
127358183	3306	5	supplier visit	830.8171428571429	2.1429
127808366	3896	1	sales call	1289.1675	3.0000
127808366	4136	2	conference	2485.6099999999997	2.7500
127808366	2885	3	site visit	2195.4049999999997	5.0000
127808366	3175	4	off-site meeting	2267.125	3.5000
132451671	2941	1	sales call	1552.32	3.0000

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Query 2: Calculating department wise total expenses, total trips, and average expense.

```
SELECT datawarehouse.dept, dept_codes.dept_name, SUM(trip_expense) AS total_dept_expense,
COUNT(dept) AS total_dept_trips, AVG(trip_expense) AS average_dept_expense
FROM datawarehouse
JOIN dept_codes
ON datawarehouse.dept = dept_codes.dept_id
GROUP BY datawarehouse.dept
ORDER BY total_dept_expense DESC;
```



dept	dept_name	total_dept_expense	total_dept_trips	average_dept_expense
204	Sales	430576.9899999998	379	1136.0870448548808
205	Purchasing	207729.42999999993	167	1243.8888023952093
203	Quality	194294.54000000004	106	1832.9673584905663
208	Executive	179188.05999999994	47	3812.5119148936155
206	Logistics	112591.99999999997	57	1975.2982456140346
207	Human Resources	81119.80999999998	33	2458.17606060606
201	Operations	53674.38	33	1626.4963636363636
200	Finance	35197.32000000001	33	1066.5854545454547
202	Information Technology	17490.600000000002	14	1249.3285714285716

Query 3 : To find out the reason for which a department takes the maximum trips

First, calculating the frequency of each reason for which each department takes trips (which will be included as a sub query for this solution)

```
SELECT dept, dept_reason_frequency.reason_code, no_of_trips, reason_description
FROM reason_codes
RIGHT JOIN (SELECT dept, reason_code, (COUNT(reason_code)) AS no_of_trips
FROM datawarehouse
GROUP BY dept, reason_code
ORDER BY dept, no_of_trips DESC) AS dept_reason_frequency
ON reason_codes.reason_code = dept_reason_frequency.reason_code;
```

```
155
156 •  SELECT dept, dept_reason_frequency.rea
157    FROM reason_codes
158    RIGHT JOIN (SELECT dept, reason_code,
159    FROM datawarehouse
```

Result Grid | Filter Rows: | Export:

dept	reason_code	no_of_trips	reason_description
200	5	8	supplier visit
200	4	8	off-site meeting
200	2	6	conference
200	1	6	sales call
200	3	5	site visit
201	3	12	site visit
201	4	10	off-site meeting
201	1	4	sales call
201	2	4	conference
201	5	3	supplier visit
202	3	5	site visit
202	4	4	off-site meeting
202	2	3	conference
202	1	2	sales call
203	2	35	conference
203	3	31	site visit
203	1	15	sales call
203	5	13	supplier visit
203	4	12	off-site meeting

We will use the above query to find reason for which each department takes maximum trips.

```
SELECT dept, max_reason_freq, temp.reason_code, reason_description
FROM (SELECT dept, Max(no_of_trips) AS max_reason_freq, reason_code
FROM (SELECT dept, dept_reason_frequency.reason_code, no_of_trips, reason_description
FROM reason_codes
RIGHT JOIN (SELECT dept, reason_code, (COUNT(reason_code)) AS no_of_trips
FROM datawarehouse
GROUP BY dept, reason_code
ORDER BY dept, no_of_trips DESC) AS dept_reason_frequency
ON reason_codes.reason_code = dept_reason_frequency.reason_code) AS reason_frequency
GROUP BY dept) AS temp
JOIN reason_codes
ON reason_codes.reason_code = temp.reason_code
ORDER BY dept;
```

	dept	max_reason_freq	reason_code	reason_description
▶	200	8	5	supplier visit
	201	12	3	site visit
	202	5	3	site visit
	203	35	2	conference
	204	226	1	sales call
	205	69	5	supplier visit
	206	19	2	conference
	207	14	2	conference
	208	13	4	off-site meeting

Query 4: Calculating total rows in the datawarehouse which have

1. settled reimbursement
2. reimbursement amount less than expense
3. not yet been reimbursed

This query adds a column to our warehouse calculating the difference between expenses from employee trips and reimbursement amounts (which will be used as a subquery for our solution)

```
SELECT datawarehouse.employee, datawarehouse.dept, datawarehouse.trip_id,
datawarehouse.reason_code, datawarehouse.account_no,
datawarehouse.trip_expense, datawarehouse.reimbursement_amount,
datawarehouse.trip_expense - datawarehouse.reimbursement_amount AS difference,
   CASE
   WHEN datawarehouse.trip_expense - datawarehouse.reimbursement_amount > 0 AND
datawarehouse.reimbursement_amount > 0 THEN 'reimbursement_amount less than trip expense'
   WHEN datawarehouse.trip_expense - datawarehouse.reimbursement_amount = 0  THEN 'amount
settled'
   WHEN datawarehouse.trip_expense - datawarehouse.reimbursement_amount > -1 AND
datawarehouse.trip_expense - datawarehouse.reimbursement_amount < 0 THEN 'amount settled'
   ELSE 'reimbursement_amount pending'
   END AS reimbursement_status
FROM datawarehouse
ORDER BY difference DESC, trip_expense DESC;
```

employee	dept	trip_id	reason_code	account_no	trip_expense	reimbursement_amount	difference	reimbursement_status
557696451	208	5264	4	1000	5913.28	0	5913.28	reimbursement_amount pending
557696451	208	4279	3	1000	5825.05	0	5825.05	reimbursement_amount pending
247505332	208	5102	4	1000	5683.07	0	5683.07	reimbursement_amount pending
995619281	206	6659	4	1000	5212.38	0	5212.38	reimbursement_amount pending
557696451	208	2989	2	2000	4903.57	0	4903.57	reimbursement_amount pending
557696451	208	6798	4	1000	4889.370000000001	0	4889.370000000001	reimbursement_amount pending
477501922	208	4404	2	1000	4703.11	0	4703.11	reimbursement_amount pending
180604167	208	5389	2	1000	4334.67	0	4334.67	reimbursement_amount pending
303083671	208	6211	3	1000	4253.79	0	4253.79	reimbursement_amount pending
247505332	208	4619	3	1000	4224.06	0	4224.06	reimbursement_amount pending
557696451	208	7107	5	1000	4110.33	0	4110.33	reimbursement_amount pending
247505332	208	3021	3	1000	4075.7699999999995	0	4075.7699999999995	reimbursement_amount pending
477501922	208	4340	3	1000	4018.7	0	4018.7	reimbursement_amount pending
103489296	204	4566	2	1000	4010.0000000000005	0	4010.0000000000005	reimbursement_amount pending
180604167	208	6870	5	1000	3992.1000000000004	0	3992.1000000000004	reimbursement_amount pending
804757664	201	3510	4	3000	3958.07	0	3958.07	reimbursement_amount pending
631676799	205	4910	4	1000	3826.56	0	3826.56	reimbursement_amount pending

The query above is being used in the following query to calculate the total rows which have settled balances, rows which were reimbursed lesser amount than expenses and rows which are yet to be reimbursed.

```
SELECT reimbursement_status, COUNT(reimbursement_status)
FROM (SELECT datawarehouse.employee, datawarehouse.dept, datawarehouse.trip_id,
datawarehouse.reason_code, datawarehouse.account_no,
datawarehouse.trip_expense, datawarehouse.reimbursement_amount,
datawarehouse.trip_expense - datawarehouse.reimbursement_amount AS difference,
   CASE
   WHEN datawarehouse.trip_expense - datawarehouse.reimbursement_amount > 0 AND
datawarehouse.reimbursement_amount > 0 THEN 'reimbursement_amount less than trip expense'
```

```
   WHEN datawarehouse.trip_expense - datawarehouse.reimbursement_amount = 0  THEN 'amount
settled'
   WHEN datawarehouse.trip_expense - datawarehouse.reimbursement_amount > -1 AND
datawarehouse.trip_expense - datawarehouse.reimbursement_amount < 0 THEN 'amount settled'
   ELSE 'reimbursement_amount pending'
   END AS reimbursement_status
FROM datawarehouse
ORDER BY difference DESC, trip_expense DESC) AS datawarehouse_2
GROUP BY reimbursement_status;
```

reimbursement_status	COUNT(reimbursement_status)
amount settled	266
reimbursement_amount less than trip expense	166
reimbursement_amount pending	437