FINAL PROJECT REPORT

ISM 6208 – Data Warehousing

Submitted by Group Alpha

Uma Srikanth Reddy Koduru	U94125452
Sindhura Alla	U06443828
Medha Alla	U16177536
Krishna Sai Chaluvadi	U48971923
Vashishta	

Major in

BUSINESS ANALYTICS AND INFORMATION SYSTEMS

Under the guidance of

DR. DONALD BERNDT

MUMA COLLEGE OF BUSINESS

UNIVERSITY OF SOUTH FLORIDA

Table of Contents

Introduction

The dataset used in this project is from the U.S. Small Business Administration (SBA) and
It consists of historical data from 1987 to 2014. It has around 7 million rows and 28 columns.
In the Data Warehousing course (ISM 6208) a final project was done under Dr. Berndt with this dataset. But the scope and the approach for this project are completely different. In this project an OLTP system has been designed and implemented as a Data Warehouse.

Part 1: Executive Summary

As the data is huge, it becomes appropriate to use big data techniques. So, the data has been cleaned using Apache Spark (PySpark) and loaded into Oracle SQL Developer for constructing the Data Warehouse. Various analytical queries have been written for analyzing the data using the OLTP Cube design (Star schema). The data is further migrated to Tableau Desktop and the same star schema is built for visualizing the data.

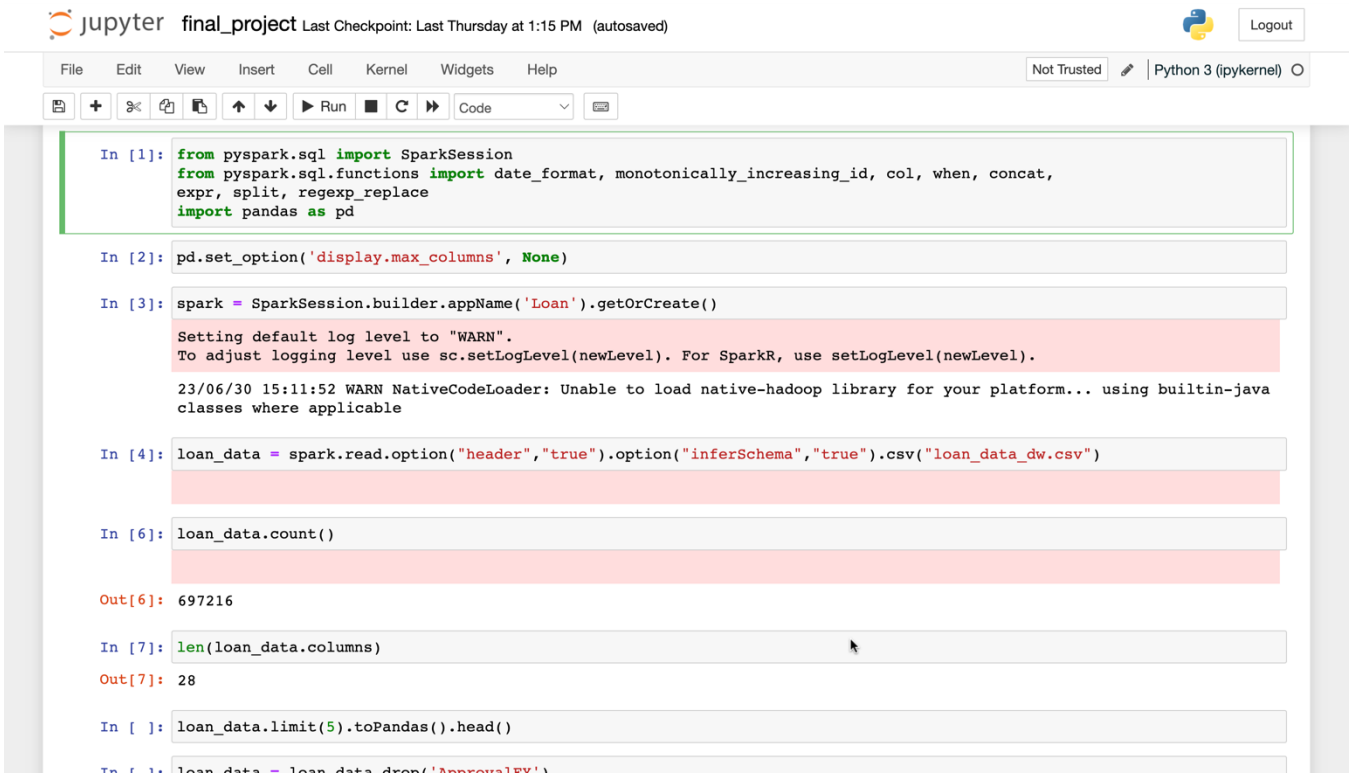
Part 2: Problem Statement

- What are the top and outstanding loans?
- What is the industry wise employee generation?
- Is there any correlation between employment generated and retained based on industries and years?

Part 3: Data Collection and Preparation

Section 3.1 Exploratory Data Analysis (EDA)

As the data has millions of records it became obvious to use big data tech like Apache Spark instead of using a pandas data frame for cleaning the data. First the data is loaded into a spark object and all the existing columns are analyzed. It is observed that there are many columns which have 0 and 1 as code for yes or no and approved or not approved. So, following the Kimball's best practices in data warehousing the values are converted to elaborate texts (used in the dimensional tables). Some of the columns in the fact tables contain the comma marks which are read as texts in oracle, so these columns are transformed using regexp_replace function and converted into an 'int' data type in the spark program. The column names have also been changed to make it into a meaningful one (used during slicing dicing operations)



Part 4: Datawarehouse Dimensional Modelling

Section 4.1 Requirement Gathering

Every system is designed based on the end goal and end user in mind. The present model is designed for a data engineer as the end user (writing analytical queries for obtaining the results).

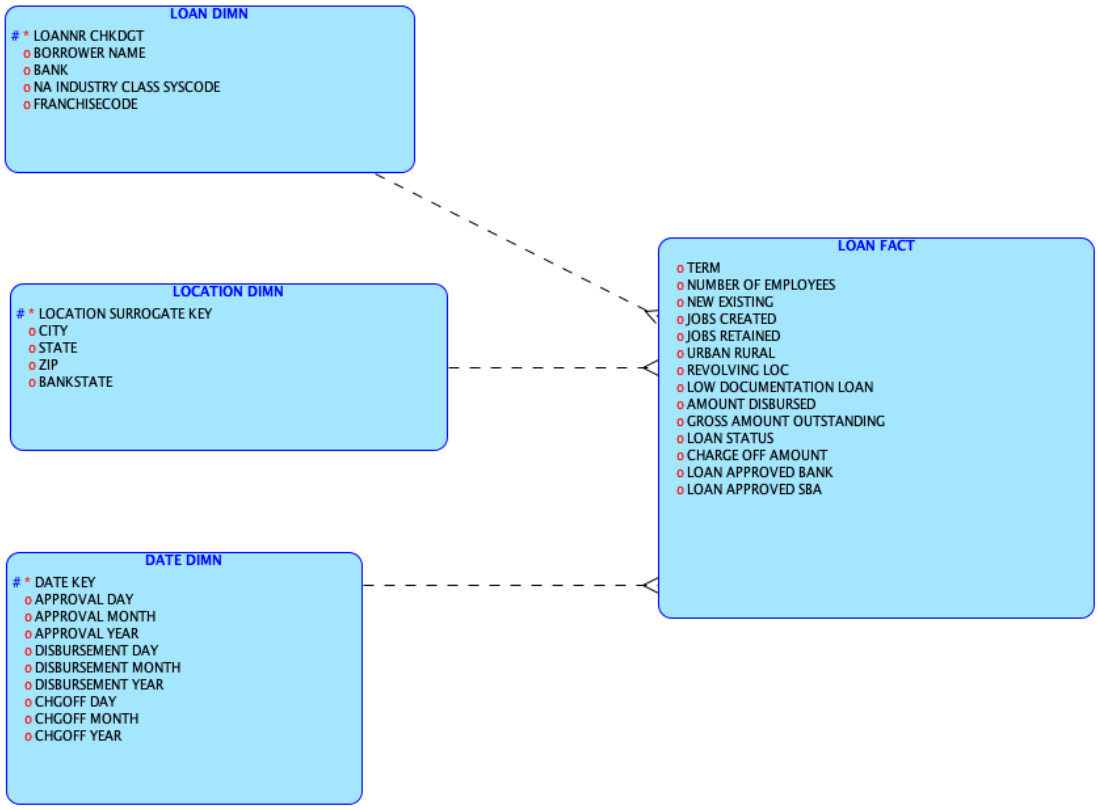
Section 4.2 Grain Identification

A very large grain may not be helpful in answering the questions in the problem statement and so a very fine grain (with seconds as time stamp) eats away the computing resources. So a grain which has a year, loan number and city is identified for the present data warehouse. This approach may be helpful in future to run some more deeper analytical queries without compromising the system performance.

Section 4.3 Dimensions and Facts

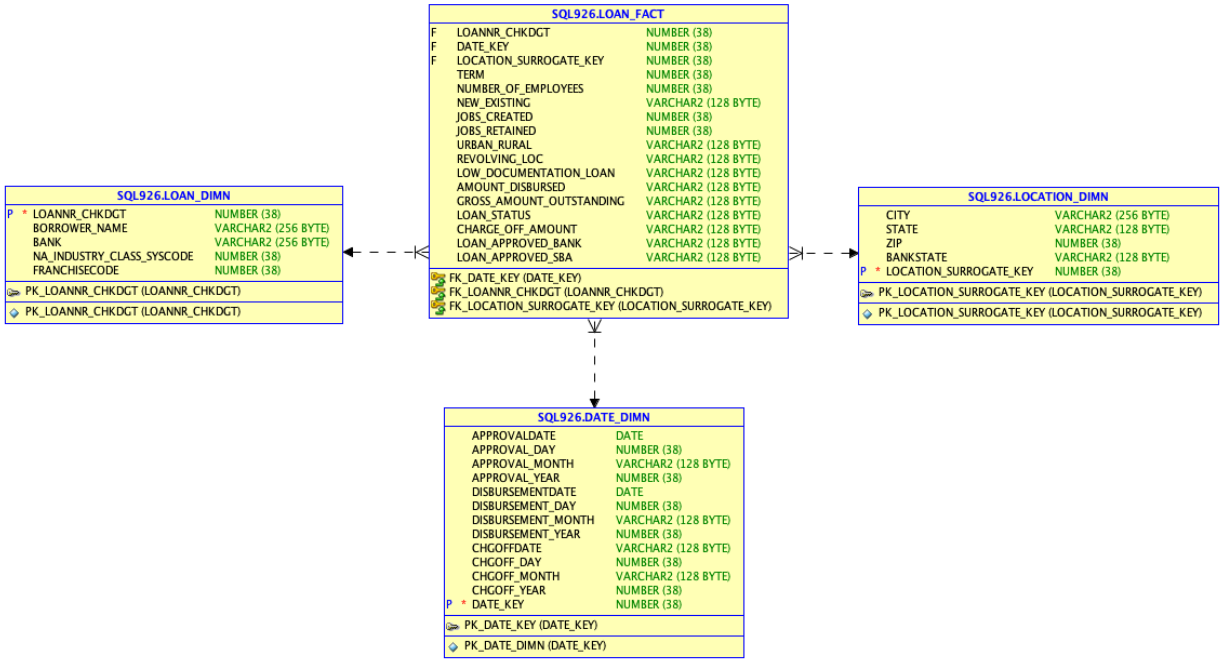
As a best practice first dimensions are identified. The dimensions identified are Location, Date and Loan dimension. A surrogate key is generated for each of the dimensions. In case of the date dimension various date columns (approval, disbursement, charged off) are concatenated to bring a unique values column for each row of the data. In the case of location dimension a unique number is generate for each row.

A single fact table is considered for this data warehouse with additive fact columns ad flag columns (transformed prior using Apache spark).


LOAN DIMN
* LOANNR CHKDGT
o BORROWER NAME
o BANK
o NA INDUSTRY CLASS SYSCODE
o FRANCHISECODE
LOCATION DIMN
* LOCATION SURROGATE KEY
o CITY
o STATE
o ZIP
o BANKSTATE
DATE DIMN
* DATE KEY
o APPROVAL DAY
o APPROVAL MONTH
o APPROVAL YEAR
o DISBURSEMENT DAY
o DISBURSEMENT MONTH
o DISBURSEMENT YEAR
o CHGOFF DAY
o CHGOFF MONTH
o CHGOFF YEAR
LOAN FACT
o TERM
o NUMBER OF EMPLOYEES
o NEW EXISTING
o JOBS CREATED
o JOBS RETAINED
o URBAN RURAL
o REVOLVING LOC
o LOW DOCUMENTATION LOAN
o AMOUNT DISBURSED
o GROSS AMOUNT OUTSTANDING
o LOAN STATUS
o CHARGE OFF AMOUNT
o LOAN APPROVED BANK
o LOAN APPROVED SBA

Part 5: Data Cube vs OLTP design

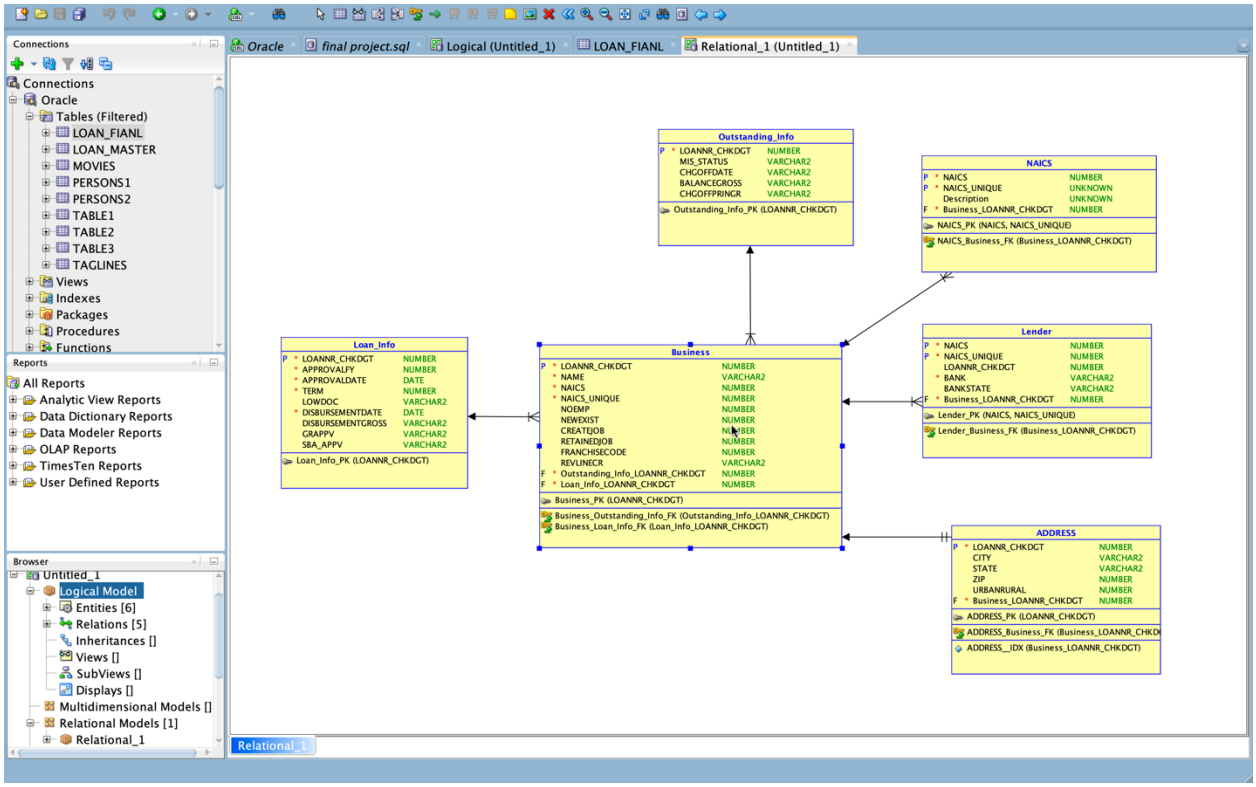
Section 5.1 Star Schema



The fact table is surrounded by the three dimensions and these fact and dimensions are related to each other by one-to-many relations.

Section 5.2 Comparison with OLTP

There is no concept of fact and dimension in the OLTP design rather the data is divided into tables based on the normalization constraints in mind and business use case. This design is not suitable for running extensive analytical queries are int would involve complex join operations.



Section 5.3 Integrity Constraints

The dimension tables are associated with an individual primary key (surrogate keys) despite some of the dimensions are having a natural key. These primary keys are referenced to the fact table using the foreign key for the respective primary key column of dimension in the fact table. Null values and redundant data is removed before assigning a particular column as primary key.

The code for this operation is given below.

```
/*Deleting the null keys from the column to make it into a primary key*/
DELETE FROM DATE_DIMN
WHERE DATE_KEY IS NULL;
/*Deleting duplicate values and preserving one value among them with lowest rowid*/
DELETE FROM DATE_DIMN
WHERE DATE_KEY IN (
  SELECT DATE_KEY
```

```
  FROM DATE_DIMN
  GROUP BY DATE_KEY
  HAVING COUNT(*) > 1
)
AND ROWID NOT IN (
  SELECT MIN(ROWID)
  FROM DATE_DIMN
  GROUP BY DATE_KEY
  HAVING COUNT(*) > 1
);
/*Adding the primary key constraint*/
ALTER TABLE DATE_DIMN
ADD CONSTRAINT PK_DATE_KEY PRIMARY KEY (DATE_KEY);


/*The above mentioned process is followed for the LOAN_DIMN table*/
DELETE FROM LOAN_DIMN
WHERE LOANNR_CHKDGT IS NULL;

DELETE FROM LOAN_DIMN
WHERE LOANNR_CHKDGT IN (
  SELECT LOANNR_CHKDGT
  FROM LOAN_DIMN
  GROUP BY LOANNR_CHKDGT
  HAVING COUNT(*) > 1
)
AND ROWID NOT IN (
  SELECT MIN(ROWID)
  FROM LOAN_DIMN
  GROUP BY LOANNR_CHKDGT
  HAVING COUNT(*) > 1
);

ALTER TABLE LOAN_DIMN
ADD CONSTRAINT PK_LOANNR_CHKDGT PRIMARY KEY (LOANNR_CHKDGT);
```

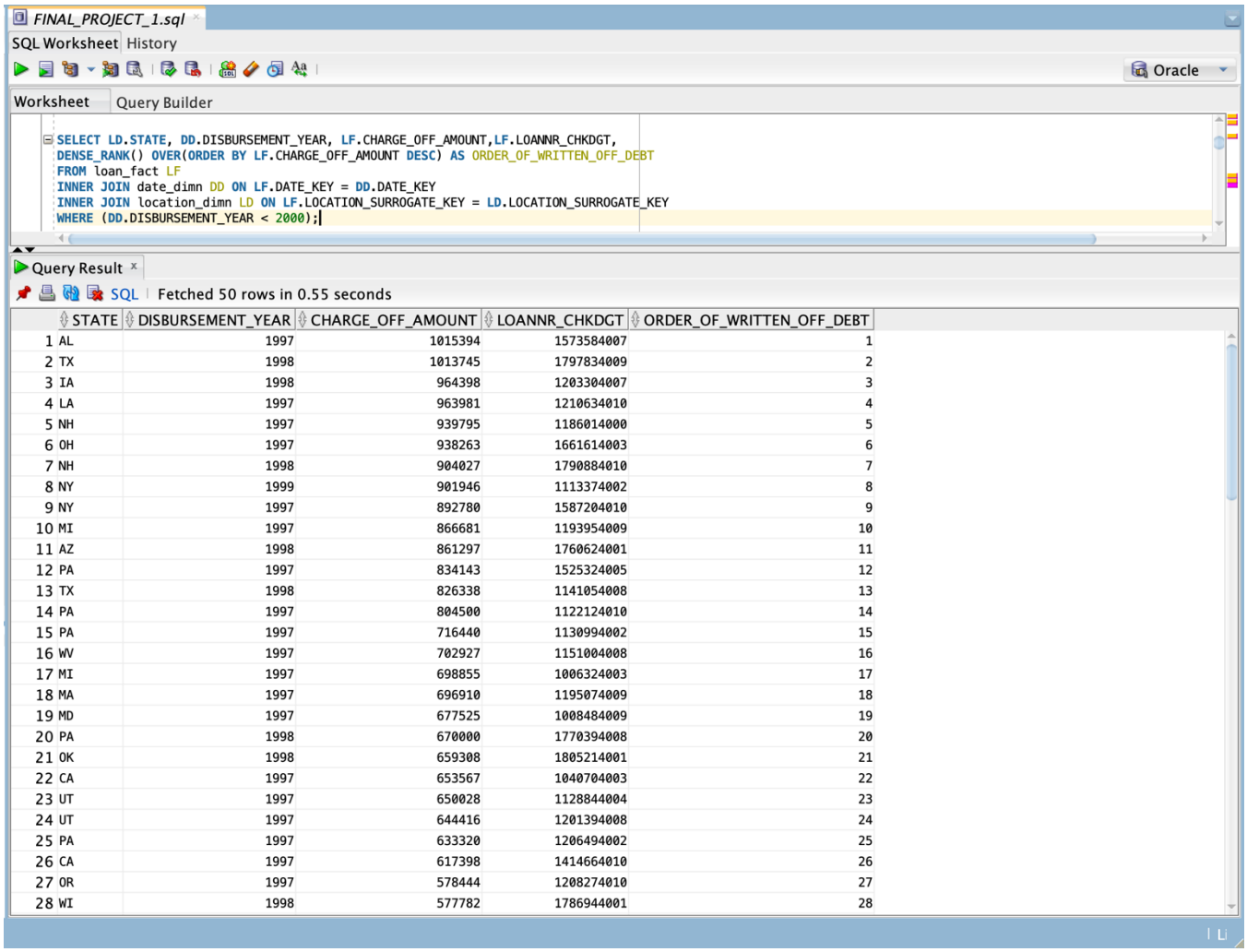
```
/*The above mentioned process is followed for the LOCATION_DIMN table*/
DELETE FROM LOCATION_DIMN
WHERE LOCATION_SURROGATE_KEY IS NULL;

DELETE FROM LOCATION_DIMN
WHERE LOCATION_SURROGATE_KEY IN(
 SELECT LOCATION_SURROGATE_KEY
 FROM LOCATION_DIMN
 GROUP BY LOCATION_SURROGATE_KEY
 HAVING COUNT(*)>1)

AND ROWID NOT IN(
 SELECT MIN(ROWID)
 FROM LOCATION_DIMN
 GROUP BY LOCATION_SURROGATE_KEY
 HAVING COUNT(*)>1);

ALTER TABLE LOCATION_DIMN
ADD CONSTRAINT PK_LOCATION_SURROGATE_KEY PRIMARY KEY(LOCATION_SURROGATE_KEY);

/*Deleting the rows with null in loan number from Loan_Fact table*/
DELETE FROM LOAN_FACT
WHERE LOANNR_CHKDGT IS NULL;

/*Deleting the row in fact table for which the referenced rows do not match with the primary key in the dimension
tables*/
DELETE FROM LOAN_FACT
WHERE DATE_KEY NOT IN (SELECT DATE_KEY FROM DATE_DIMN);
/*Adding the foreign key constraint*/
ALTER TABLE LOAN_FACT
ADD CONSTRAINT FK_DATE_KEY FOREIGN KEY(DATE_KEY) REFERENCES DATE_DIMN(DATE_KEY);

/*Data cleaning by deleting the attributes that do not contribute to the business problem*/
DELETE FROM LOAN_FACT
WHERE LOANNR_CHKDGT NOT IN (SELECT LOANNR_CHKDGT FROM LOAN_DIMN);
```

```
ALTER TABLE LOAN_FACT
ADD CONSTRAINT FK_LOANNR_CHKDGT FOREIGN KEY(LOANNR_CHKDGT) REFERENCES
LOAN_DIMN(LOANNR_CHKDGT);

/**/
DELETE FROM LOAN_FACT
WHERE LOCATION_SURROGATE_KEY NOT IN (SELECT LOCATION_SURROGATE_KEY FROM
LOCATION_DIMN);

ALTER TABLE LOAN_FACT
ADD CONSTRAINT FK_LOCATION_SURROGATE_KEY FOREIGN KEY(LOCATION_SURROGATE_KEY)
REFERENCES LOCATION_DIMN(LOCATION_SURROGATE_KEY);
```

Part 6: Analytics

Section 6.1 Online Analytical Processing

DENSE RANK:



FINAL_PROJECT_1.sql

SQL Worksheet | History

Oracle

Worksheet | Query Builder

```
SELECT LD.STATE, DD.DISBURSEMENT_YEAR, LF.CHARGE_OFF_AMOUNT,LF.LOANNR_CHKDGT,
DENSE_RANK() OVER(ORDER BY LF.CHARGE_OFF_AMOUNT DESC) AS ORDER_OF_WRITTEN_OFF_DEBT
FROM loan_fact LF
INNER JOIN date_dimn DD ON LF.DATE_KEY = DD.DATE_KEY
INNER JOIN location_dimn LD ON LF.LOCATION_SURROGATE_KEY = LD.LOCATION_SURROGATE_KEY
WHERE (DD.DISBURSEMENT_YEAR < 2000);
```

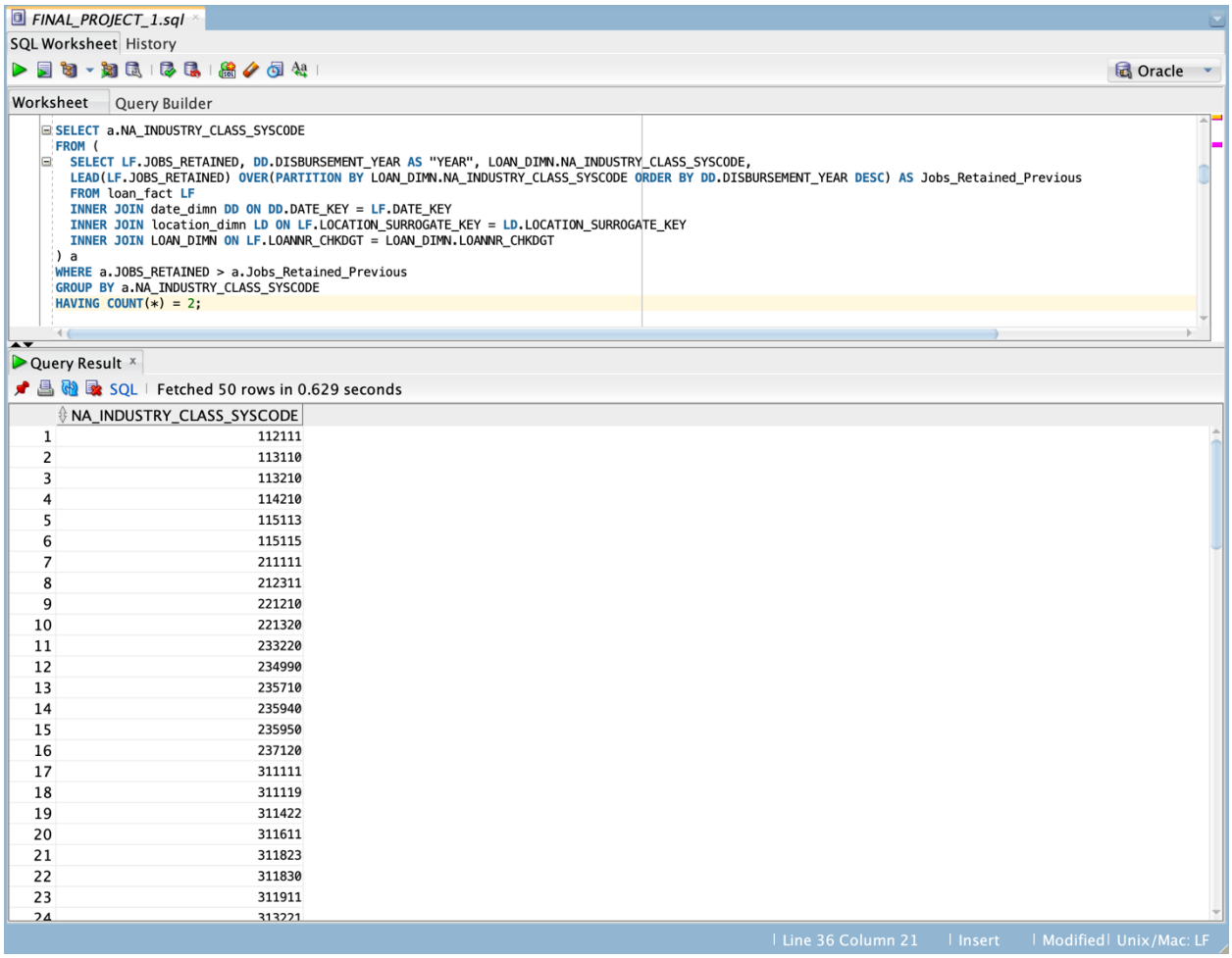
Query Result

SQL | Fetched 50 rows in 0.55 seconds

	STATE	DISBURSEMENT_YEAR	CHARGE_OFF_AMOUNT	LOANNR_CHKDGT	ORDER_OF_WRITTEN_OFF_DEBT
1	AL	1997	1015394	1573584007	1
2	TX	1998	1013745	1797834009	2
3	IA	1998	964398	1203304007	3
4	LA	1997	963981	1210634010	4
5	NH	1997	939795	1186014000	5
6	OH	1997	938263	1661614003	6
7	NH	1998	904027	1790884010	7
8	NY	1999	901946	1113374002	8
9	NY	1997	892780	1587204010	9
10	MI	1997	866681	1193954009	10
11	AZ	1998	861297	1760624001	11
12	PA	1997	834143	1525324005	12
13	TX	1998	826338	1141054008	13
14	PA	1997	804500	1122124010	14
15	PA	1997	716440	1130994002	15
16	WV	1997	702927	1151004008	16
17	MI	1997	698855	1006324003	17
18	MA	1997	696910	1195074009	18
19	MD	1997	677525	1008484009	19
20	PA	1998	670000	1770394008	20
21	OK	1998	659308	1805214001	21
22	CA	1997	653567	1040704003	22
23	UT	1997	650028	1128844004	23
24	UT	1997	644416	1201394008	24
25	PA	1997	633320	1206494002	25
26	CA	1997	617398	1414664010	26
27	OR	1997	578444	1208274010	27
28	WI	1998	577782	1786944001	28

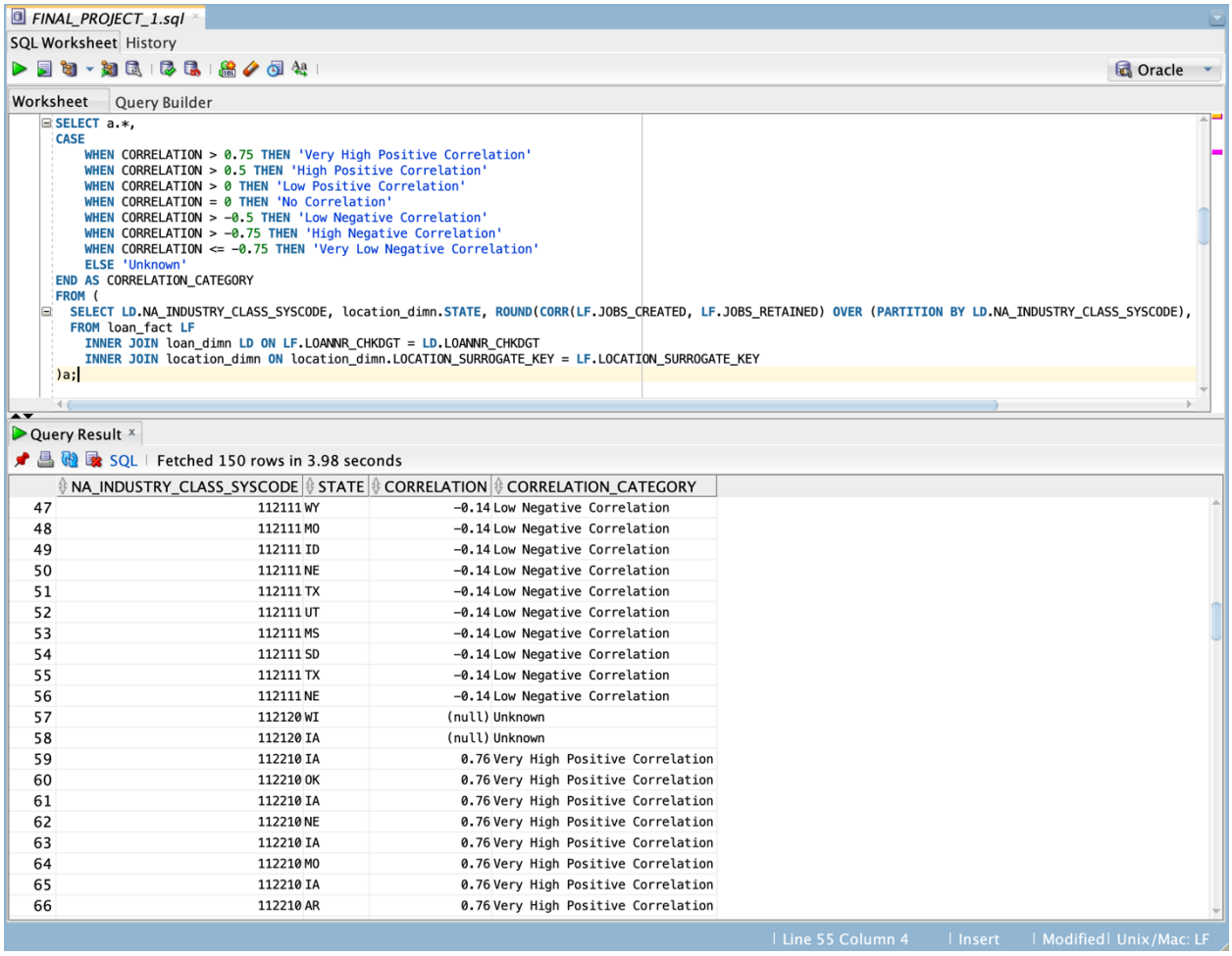
In the above analytical query, the loan numbers (Loannr_Chkdgt) are ranked based on highest written off debt by the banks using the dense_rank function. The loan numbers are also shown how much loan they have taken from the banks by state and year.

LEAD:



The above lead query returns the industries where the jobs retained are more than that of at least two consecutive previous years.

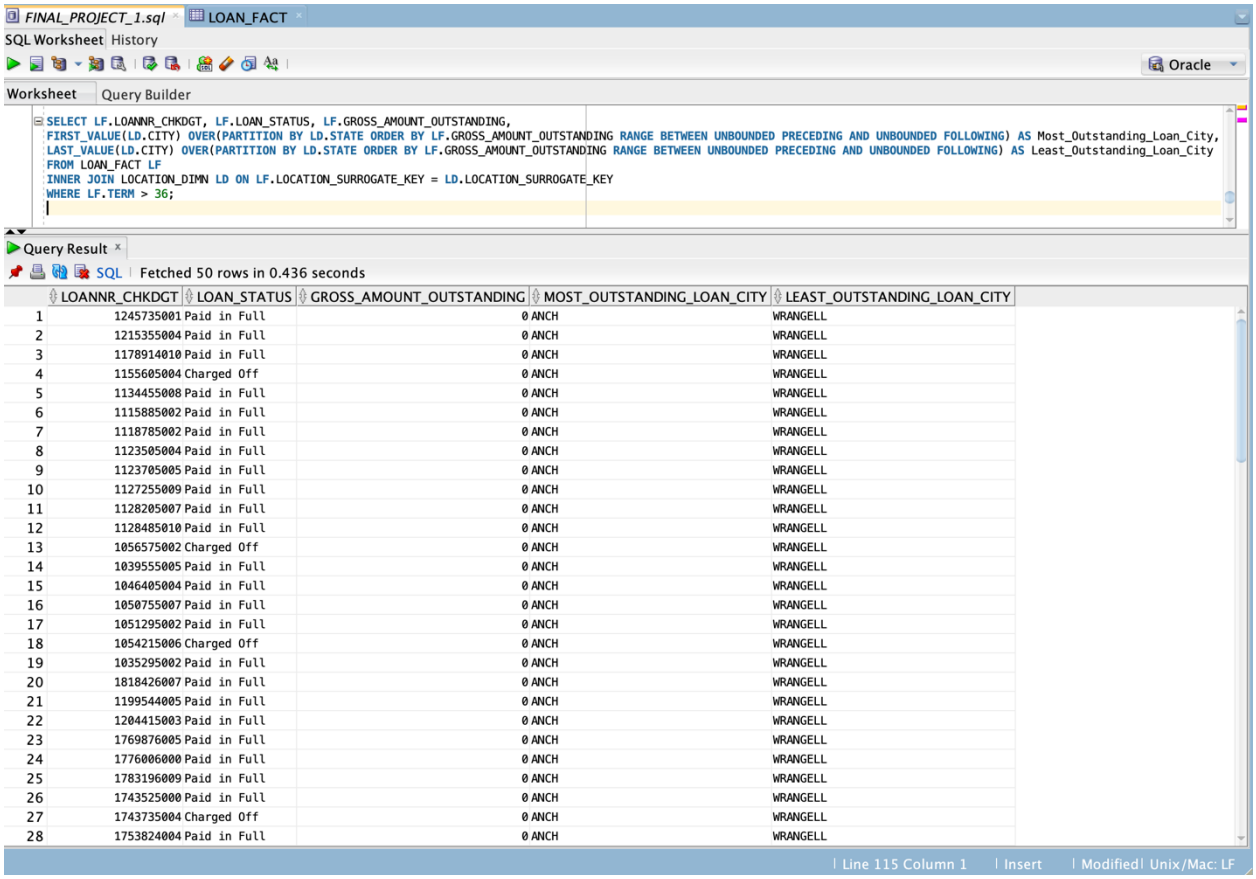
CORRELATION:



	NA_INDUSTRY_CLASS_SYSCODE	STATE	CORRELATION	CORRELATION_CATEGORY
47	112111	WY	-0.14	Low Negative Correlation
48	112111	MO	-0.14	Low Negative Correlation
49	112111	ID	-0.14	Low Negative Correlation
50	112111	NE	-0.14	Low Negative Correlation
51	112111	TX	-0.14	Low Negative Correlation
52	112111	UT	-0.14	Low Negative Correlation
53	112111	MS	-0.14	Low Negative Correlation
54	112111	SD	-0.14	Low Negative Correlation
55	112111	TX	-0.14	Low Negative Correlation
56	112111	NE	-0.14	Low Negative Correlation
57	112120	WI	(null)	Unknown
58	112120	IA	(null)	Unknown
59	112210	IA	0.76	Very High Positive Correlation
60	112210	OK	0.76	Very High Positive Correlation
61	112210	IA	0.76	Very High Positive Correlation
62	112210	NE	0.76	Very High Positive Correlation
63	112210	IA	0.76	Very High Positive Correlation
64	112210	MO	0.76	Very High Positive Correlation
65	112210	IA	0.76	Very High Positive Correlation
66	112210	AR	0.76	Very High Positive Correlation

In the above query for each of the NAICS (Industry code) the correlation between the jobs created and retained are shown by state. Each of the correlation categories are sown using case statement in the sql query.

FIRST VALUE, LAST VALUE:



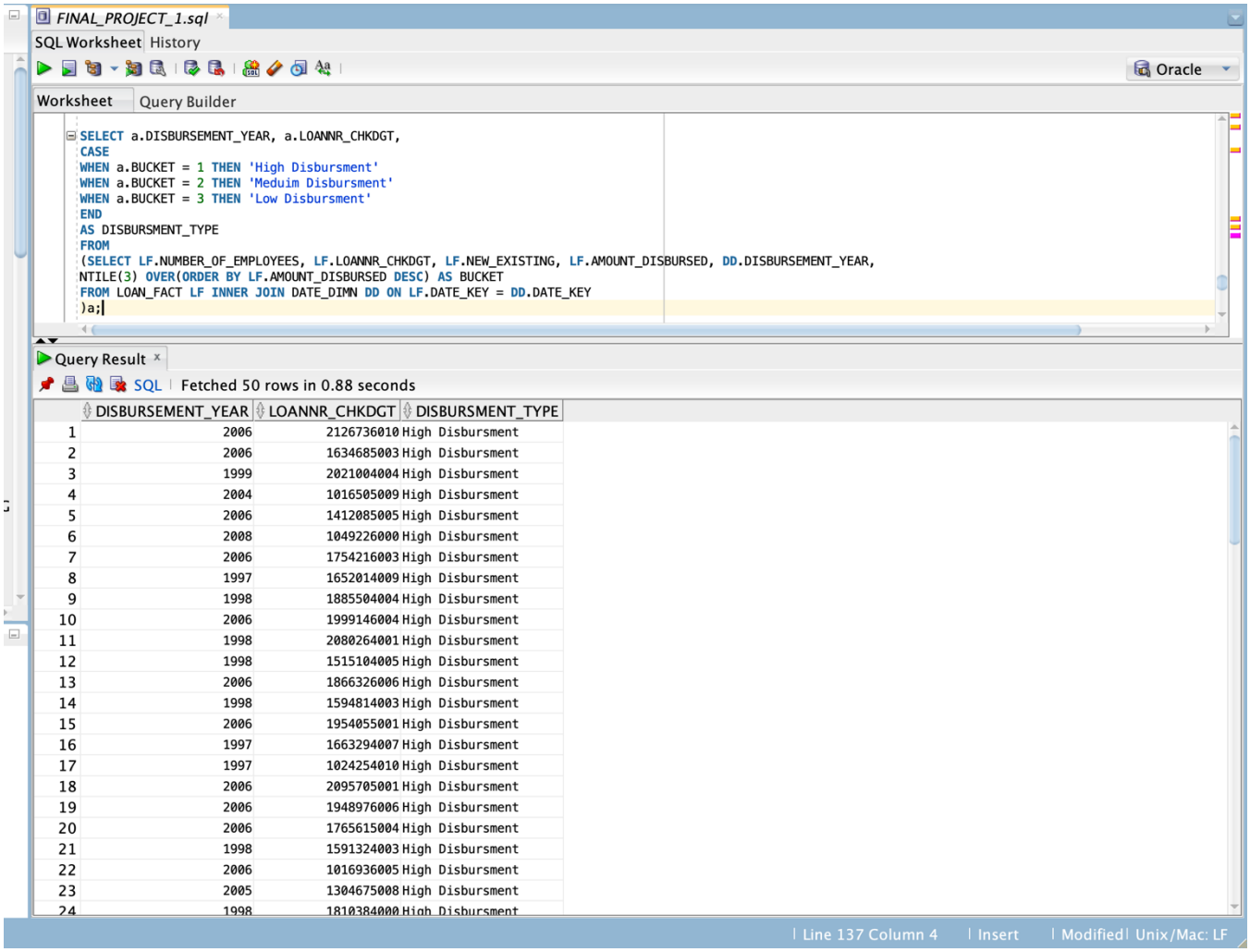
```
SELECT LF.LOANNR_CHKDGT, LF.LOAN_STATUS, LF.GROSS_AMOUNT_OUTSTANDING,
FIRST_VALUE(LD.CITY) OVER(PARTITION BY LD.STATE ORDER BY LF.GROSS_AMOUNT_OUTSTANDING RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS Most_Outstanding_Loan_City,
LAST_VALUE(LD.CITY) OVER(PARTITION BY LD.STATE ORDER BY LF.GROSS_AMOUNT_OUTSTANDING RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS Least_Outstanding_Loan_City
FROM LOAN_FACT LF
INNER JOIN LOCATION_DIMN LD ON LF.LOCATION_SURROGATE_KEY = LD.LOCATION_SURROGATE_KEY
WHERE LF.TERM > 36;
```

Fetched 50 rows in 0.436 seconds

	LOANNR_CHKDGT	LOAN_STATUS	GROSS_AMOUNT_OUTSTANDING	MOST_OUTSTANDING_LOAN_CITY	LEAST_OUTSTANDING_LOAN_CITY
1	1245735001	Paid in Full	0	ANCH	WRANGELL
2	1215355004	Paid in Full	0	ANCH	WRANGELL
3	1178914010	Paid in Full	0	ANCH	WRANGELL
4	1155605004	Charged Off	0	ANCH	WRANGELL
5	1134455008	Paid in Full	0	ANCH	WRANGELL
6	1115885002	Paid in Full	0	ANCH	WRANGELL
7	1118785002	Paid in Full	0	ANCH	WRANGELL
8	1123505004	Paid in Full	0	ANCH	WRANGELL
9	1123705005	Paid in Full	0	ANCH	WRANGELL
10	1127255009	Paid in Full	0	ANCH	WRANGELL
11	1128205007	Paid in Full	0	ANCH	WRANGELL
12	1128485010	Paid in Full	0	ANCH	WRANGELL
13	1056575002	Charged Off	0	ANCH	WRANGELL
14	1039555005	Paid in Full	0	ANCH	WRANGELL
15	1046405004	Paid in Full	0	ANCH	WRANGELL
16	1050755007	Paid in Full	0	ANCH	WRANGELL
17	1051295002	Paid in Full	0	ANCH	WRANGELL
18	1054215006	Charged Off	0	ANCH	WRANGELL
19	1035295002	Paid in Full	0	ANCH	WRANGELL
20	1818426007	Paid in Full	0	ANCH	WRANGELL
21	1199544005	Paid in Full	0	ANCH	WRANGELL
22	1204415003	Paid in Full	0	ANCH	WRANGELL
23	1769876005	Paid in Full	0	ANCH	WRANGELL
24	1776006000	Paid in Full	0	ANCH	WRANGELL
25	1783196009	Paid in Full	0	ANCH	WRANGELL
26	1743525000	Paid in Full	0	ANCH	WRANGELL
27	1743735004	Charged Off	0	ANCH	WRANGELL
28	1753824004	Paid in Full	0	ANCH	WRANGELL

In the above query the city with highest and lowest outstanding loan amount for a loan greater than 3 years (36 months) are depicted. Window function (range between unbounded preceding and unbounded following) is used.

NTILE:



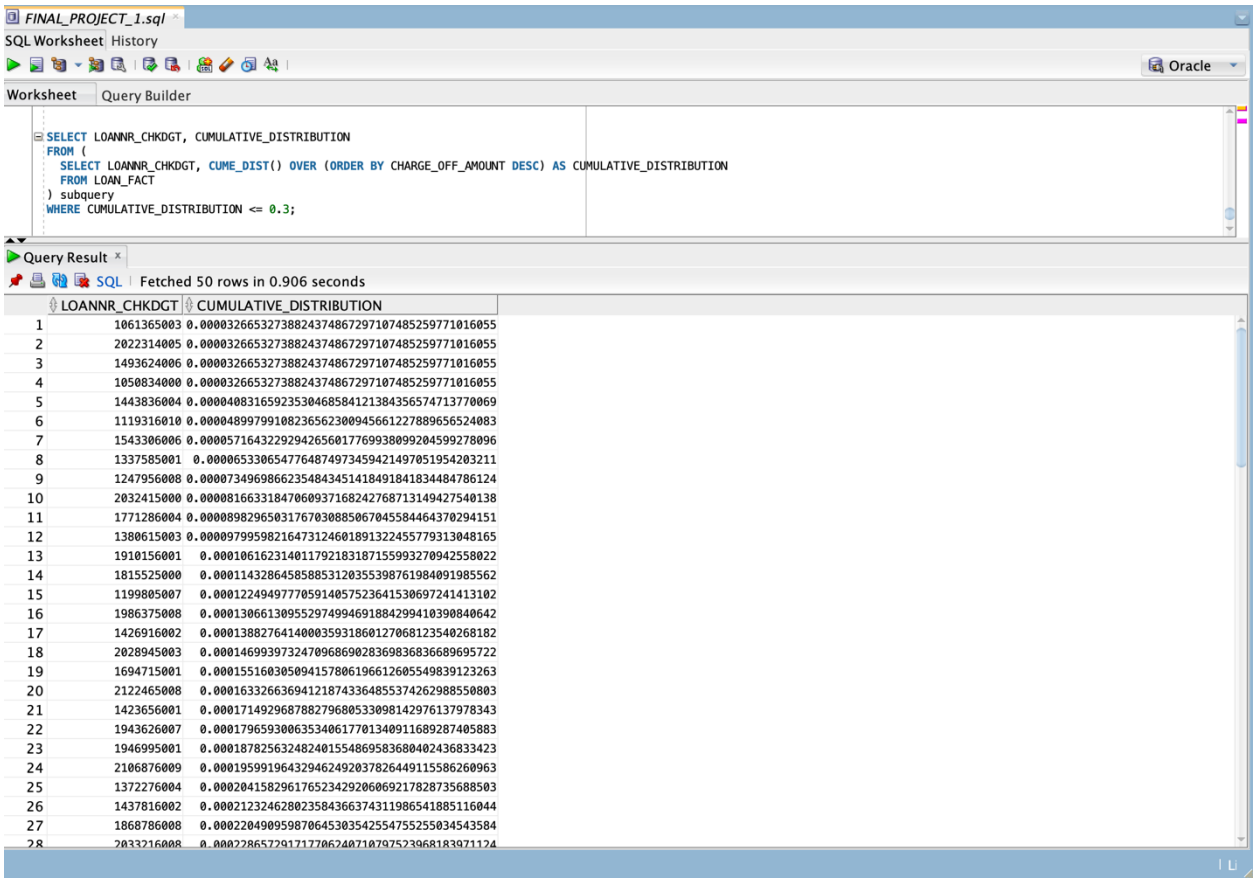
```
SELECT a.DISBURSEMENT_YEAR, a.LOANNR_CHKDGT,
CASE
WHEN a.BUCKET = 1 THEN 'High Disbursment'
WHEN a.BUCKET = 2 THEN 'Meduim Disbursment'
WHEN a.BUCKET = 3 THEN 'Low Disbursment'
END
AS DISBURSMENT_TYPE
FROM
(SELECT LF.NUMBER_OF_EMPLOYEES, LF.LOANNR_CHKDGT, LF.NEW_EXISTING, LF.AMOUNT_DISBURSED, DD.DISBURSEMENT_YEAR,
NTILE(3) OVER(ORDER BY LF.AMOUNT_DISBURSED DESC) AS BUCKET
FROM LOAN_FACT LF INNER JOIN DATE_DIMN DD ON LF.DATE_KEY = DD.DATE_KEY
)a;
```

Fetched 50 rows in 0.88 seconds

	DISBURSEMENT_YEAR	LOANNR_CHKDGT	DISBURSMENT_TYPE
1	2006	2126736010	High Disbursment
2	2006	1634685003	High Disbursment
3	1999	2021004004	High Disbursment
4	2004	1016505009	High Disbursment
5	2006	1412085005	High Disbursment
6	2008	1049226000	High Disbursment
7	2006	1754216003	High Disbursment
8	1997	1652014009	High Disbursment
9	1998	1885504004	High Disbursment
10	2006	1999146004	High Disbursment
11	1998	2080264001	High Disbursment
12	1998	1515104005	High Disbursment
13	2006	1866326006	High Disbursment
14	1998	1594814003	High Disbursment
15	2006	1954055001	High Disbursment
16	1997	1663294007	High Disbursment
17	1997	1024254010	High Disbursment
18	2006	2095705001	High Disbursment
19	2006	1948976006	High Disbursment
20	2006	1765615004	High Disbursment
21	1998	1591324003	High Disbursment
22	2006	1016936005	High Disbursment
23	2005	1304675008	High Disbursment
24	1998	1810384000	High Disbursment

In the above analytical query NTILE is used to create buckets for 3 equal sizes based on amount disbursed (High, Medium, and Low).

CUMULATIVE DISTRIBUTION:



```
FINAL_PROJECT_1.sql
SQL Worksheet  History
Oracle
Worksheet  Query Builder

SELECT LOANNR_CHKDGT, CUMULATIVE_DISTRIBUTION
FROM (
  SELECT LOANNR_CHKDGT, CUME_DIST() OVER (ORDER BY CHARGE_OFF_AMOUNT DESC) AS CUMULATIVE_DISTRIBUTION
  FROM LOAN_FACT
) subquery
WHERE CUMULATIVE_DISTRIBUTION <= 0.3;
```

Query Result — SQL | Fetched 50 rows in 0.906 seconds

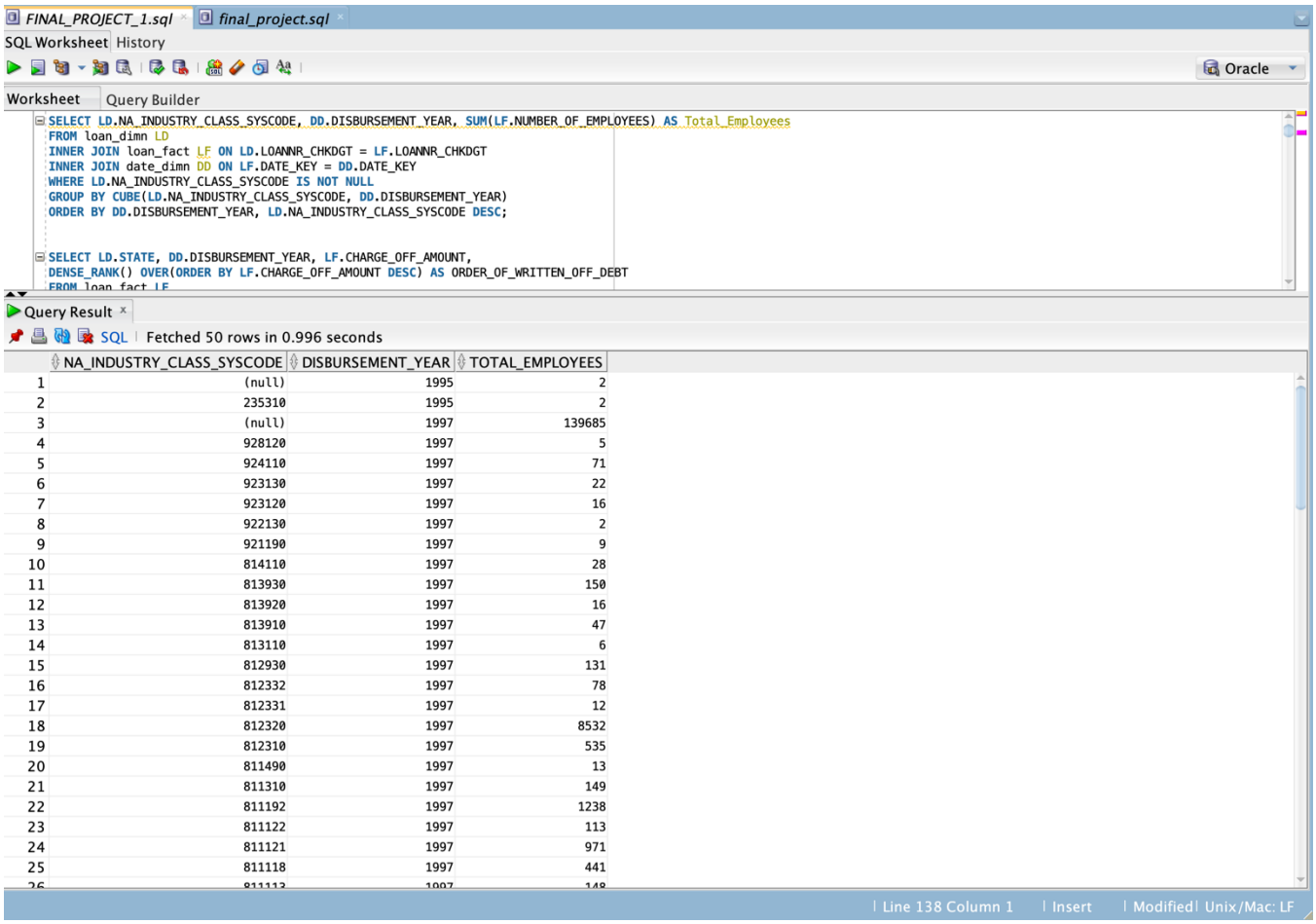
	LOANNR_CHKDGT	CUMULATIVE_DISTRIBUTION
1	1061365003	0.000032665327388243748672971074852597710160 55
2	2022314005	0.000032665327388243748672971074852597710160 55
3	1493624006	0.000032665327388243748672971074852597710160 55
4	1050834000	0.000032665327388243748672971074852597710160 55
5	1443836004	0.000040831659235304685841213843565747137700 69
6	1119316010	0.000048997991082365623009456612278896565240 83
7	1543306006	0.000057164322929426560177699380992045992780 96
8	1337585001	0.000065330654776487497345942149705195420321 1
9	1247956008	0.000073496986623548434514184918418344847861 24
10	2032415000	0.000081663318470609371682427687131494275401 38
11	1771286004	0.000089829650317670308850670455844643702941 51
12	1380615003	0.000097995982164731246018913224557793130481 65
13	1910156001	0.000106162314011792183187155993270942558022
14	1815525000	0.000114328645858853120355398761984091985562
15	1199805007	0.000122494977705914057523641530697241413102
16	1986375008	0.000130661309552974994691884299410390840642
17	1426916002	0.000138827641400035931860127068123540268182
18	2028945003	0.000146993973247096869028369836836689695722
19	1694715001	0.000155160305094157806196612605549839123263
20	2122465008	0.000163326636941218743364855374262988550803
21	1423656001	0.000171492968788279680533098142976137978343
22	1943626007	0.000179659300635340617701340911689287405883
23	1946995001	0.000187825632482401554869583680402436833423
24	2106876009	0.000195991964329462492037826449115586260963
25	1372276004	0.000204158296176523429206069217828735688503
26	1437816002	0.000212324628023584366374311986541885116044
27	1868786008	0.000220490959870645303542554755255034543584
28	2033216008	0.000228657291717706240710797523968183971124

Using cumulative distribution in the query the top 30 percent loan numbers (based on written off amount) in the existing loans are shown.

Part 7: Performance Tuning

Section 7.1 Aggregation

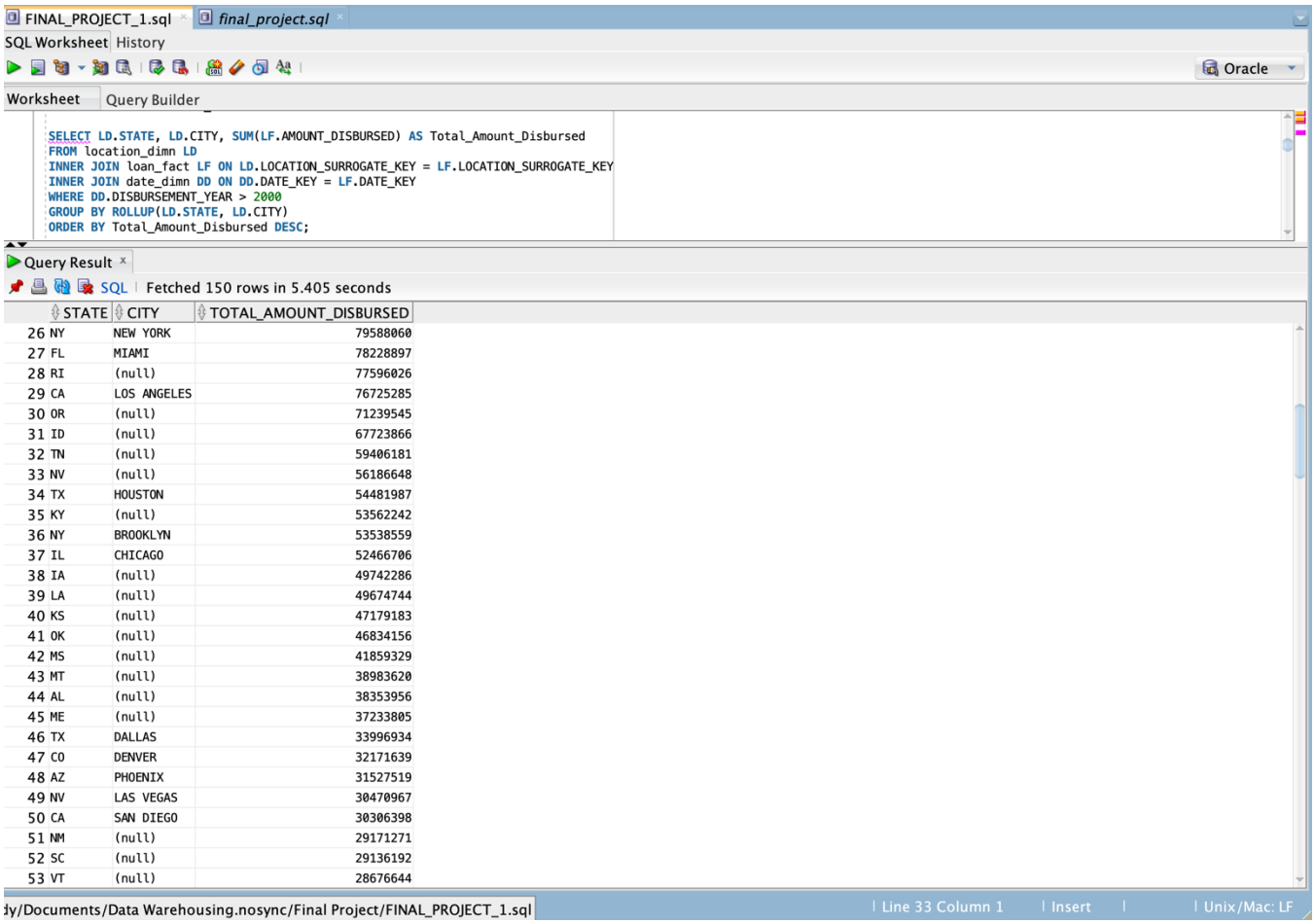
CUBE:



	NA_INDUSTRY_CLASS_SYSCODE	DISBURSEMENT_YEAR	TOTAL_EMPLOYEES
1	(null)	1995	2
2	235310	1995	2
3	(null)	1997	139685
4	928120	1997	5
5	924110	1997	71
6	923130	1997	22
7	923120	1997	16
8	922130	1997	2
9	921190	1997	9
10	814110	1997	28
11	813930	1997	150
12	813920	1997	16
13	813910	1997	47
14	813110	1997	6
15	812930	1997	131
16	812332	1997	78
17	812331	1997	12
18	812320	1997	8532
19	812310	1997	535
20	811490	1997	13
21	811310	1997	149
22	811192	1997	1238
23	811122	1997	113
24	811121	1997	971
25	811118	1997	441

In the above group by cube query the subtotal of number of employees for each of the NA_Industry_Syscode and for each of the disbursement year are calculated. Along with that a grand total of total number of employees is calculated.

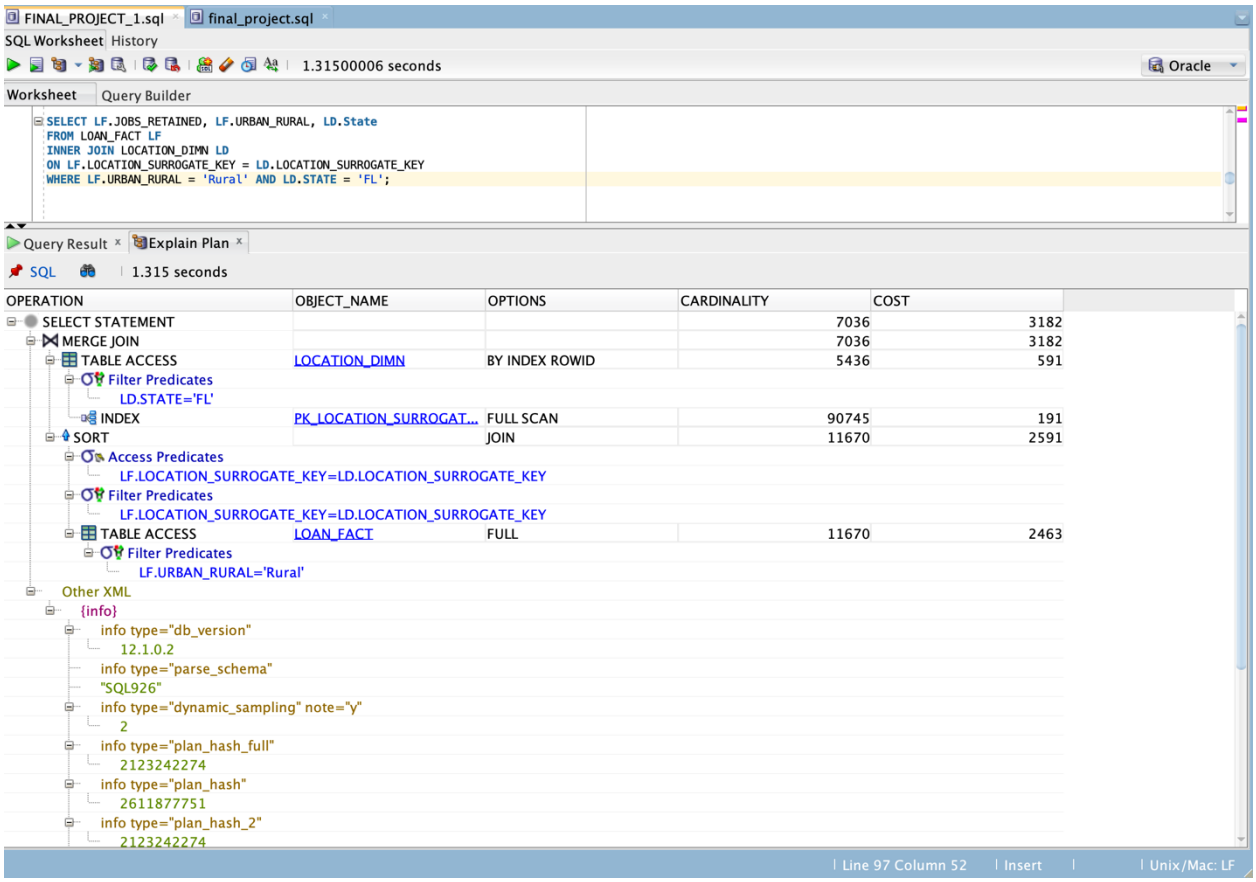
ROLLUP:



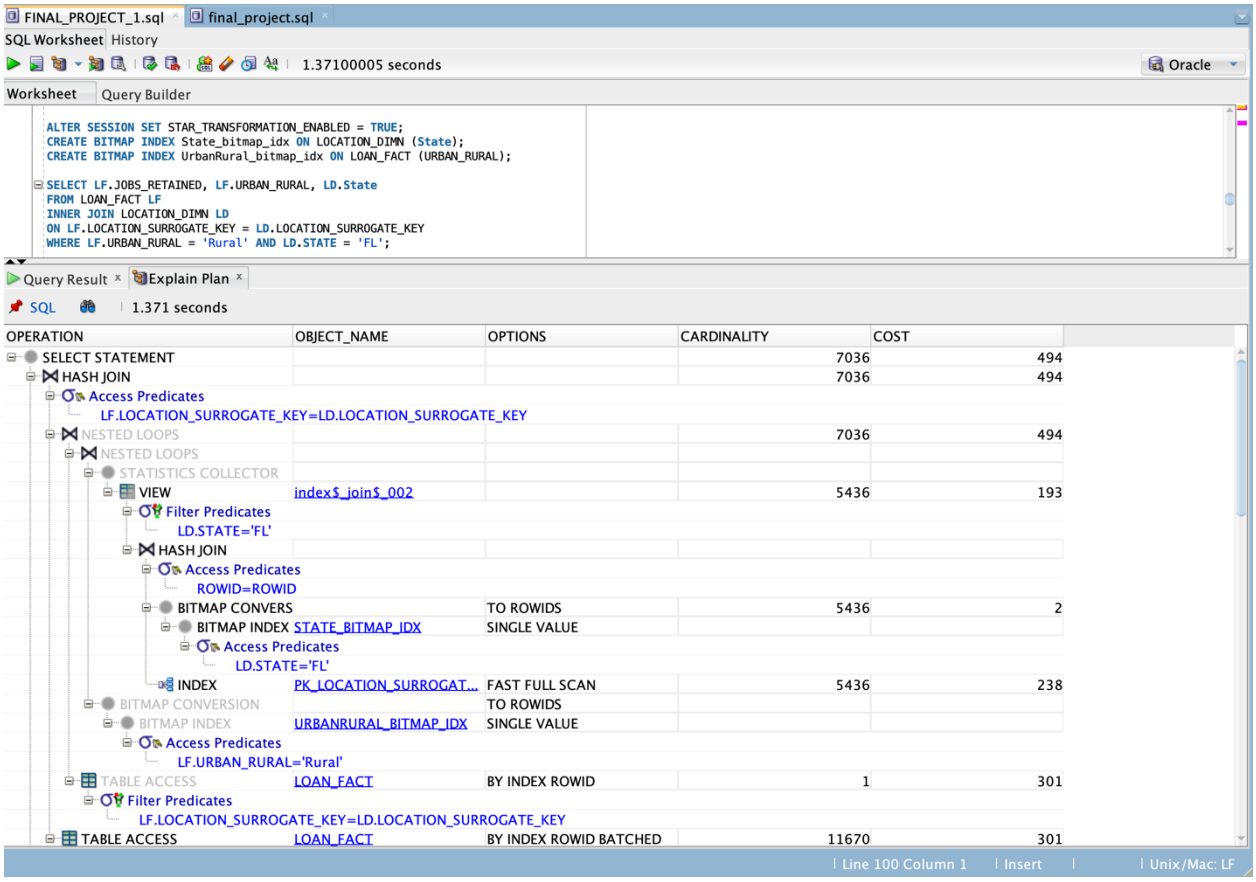
This rollup query displays subtotal of amount disbursed and grand total of amount disbursed. The sub totals are calculated based on the group by clause of state and city. In other words, the total amount disbursed per state is calculated as subtotals and per each city within the respective state is calculated. Also, the grand total i.e., total amount disbursed for all the states is calculated.

Section 7.2 Indexing

As we are dealing with star schema, the star transformation is enabled. Already the primary keys in dimension tables contain b-tree indexes (created on primary key by default) so to run the query faster where the cardinality is low (example: Flag column of Urban_ Rural has values of either Urban, Rural or Undefined) a bitmap index is applied on the columns of Uban_Rural on Loan_Fact table and State column in Location_Dimenison Table.



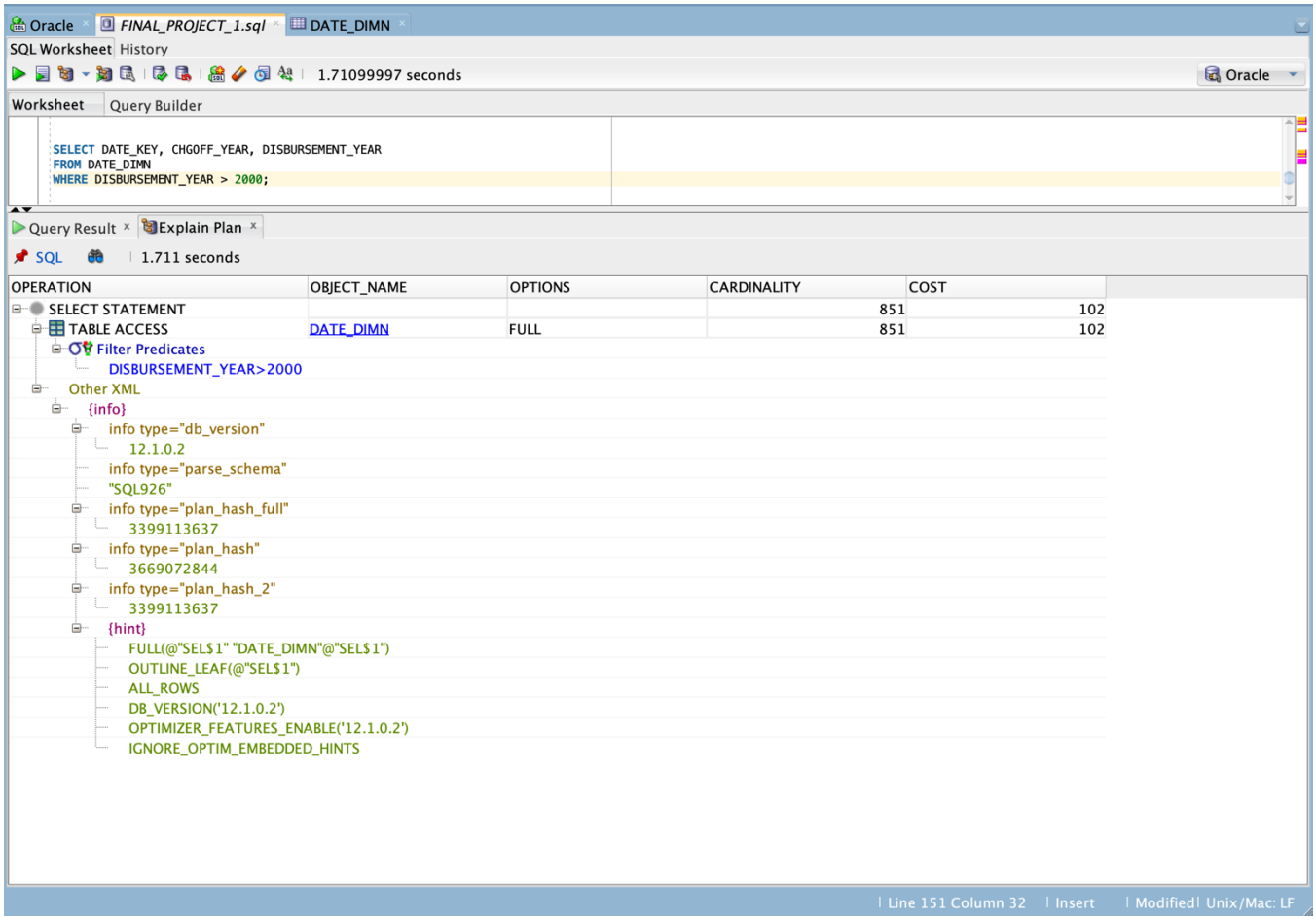
The cost of the query involving both the columns in the join query is 3182.



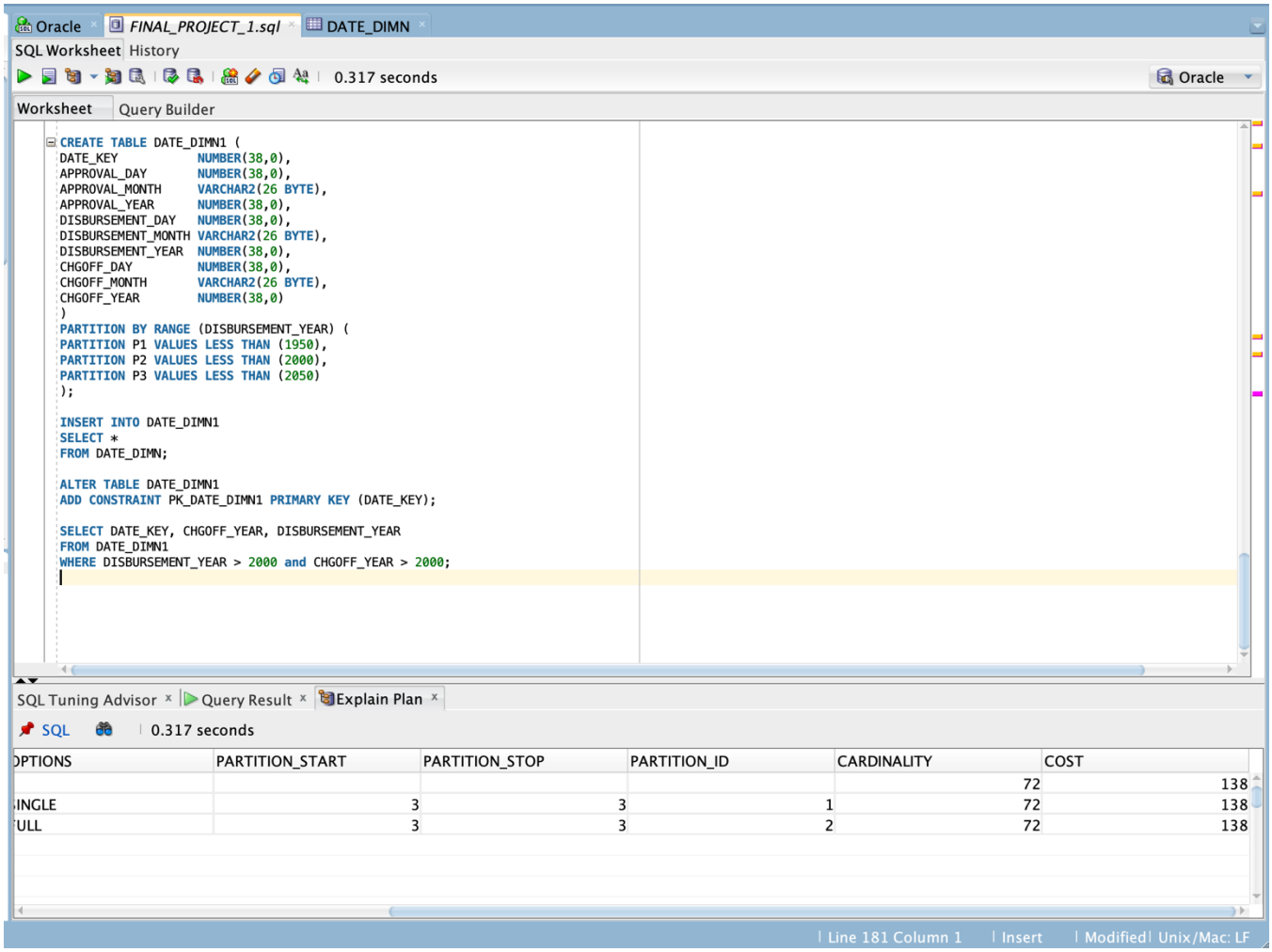
It is observed adding bitmap indexes in the flag columns of Fact table and Low cardinality columns in the dimensional has significantly improve the query cost bringing it down to 494 from 3182.

Section 7.3 Partitioning

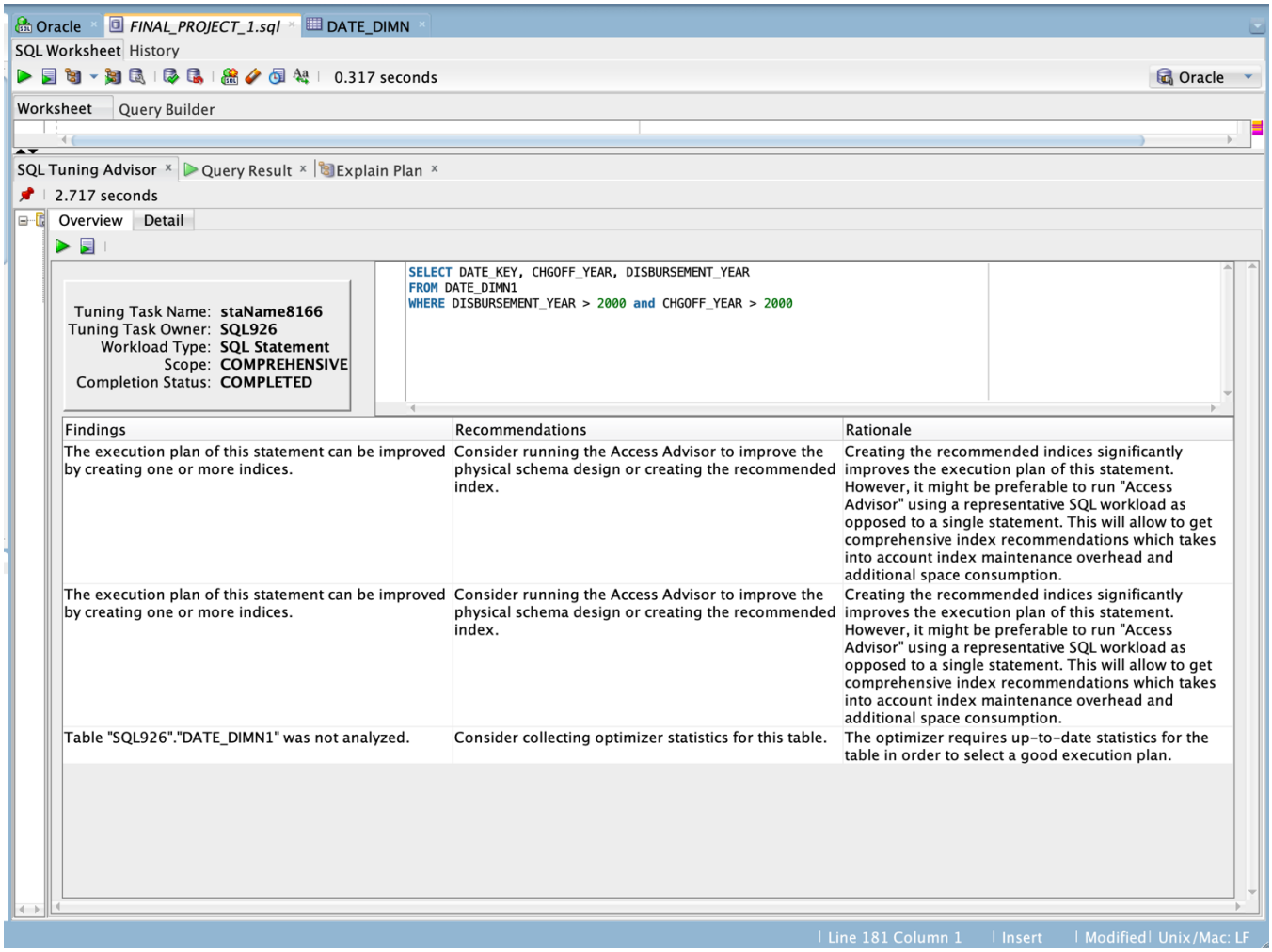
A query on DATE_DIMN table is run that involves the Disbursment_Year column and the query cost is shown below



Now a new table is created DATE_DIMN1 with range partitioned column on Disbursment_Year, now the query cost is as shown below. It is observed that the query cost has increased to our surprise despite partitioning the column.



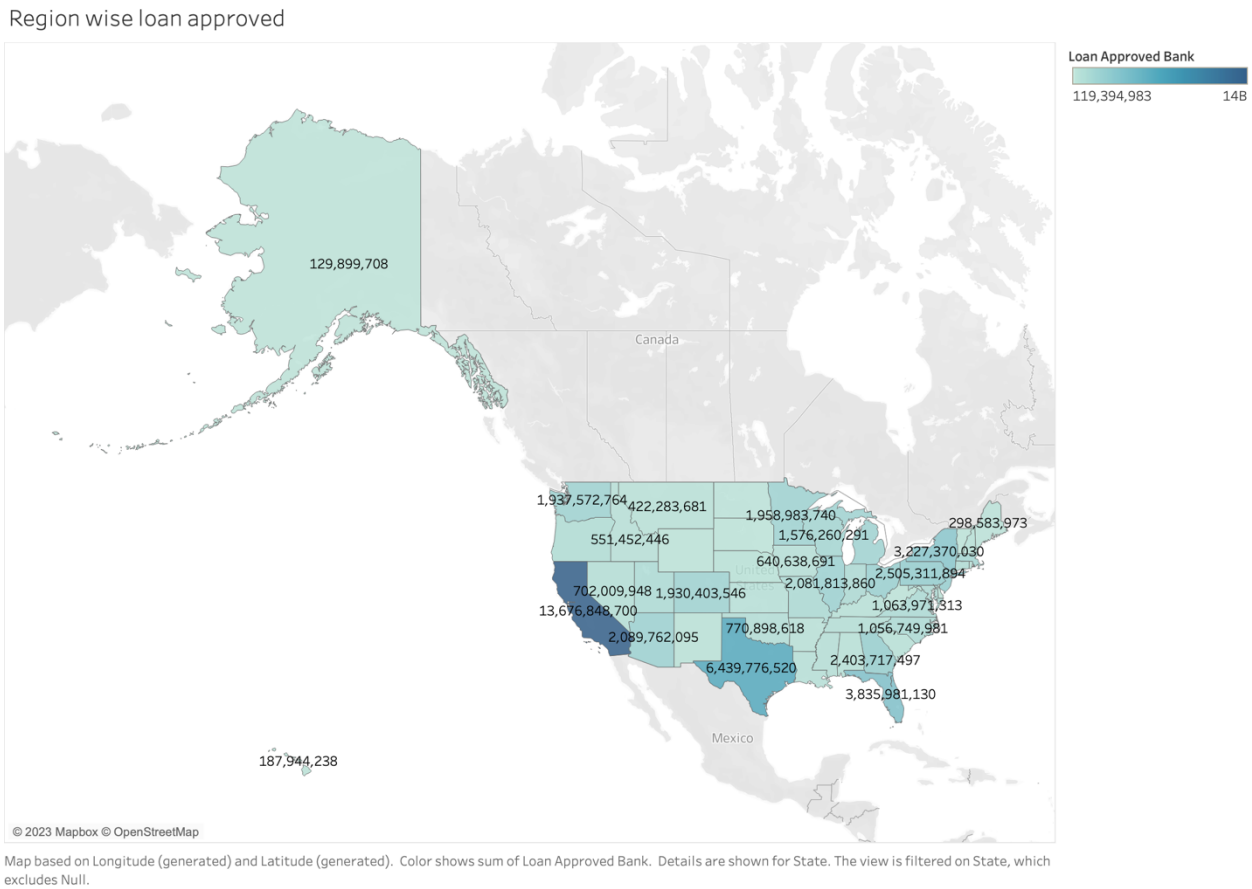
So Sql Tuning advisor is run to get the following recommendations.


Oracle
FINAL_PROJECT_1.sql
DATE_DIMN
SQL Worksheet
History
0.317 seconds
Oracle
Worksheet
Query Builder
SQL Tuning Advisor
Query Result
Explain Plan
2.717 seconds
Overview
Detail
Tuning Task Name: staName8166
Tuning Task Owner: SQL926
Workload Type: SQL Statement
Scope: COMPREHENSIVE
Completion Status: COMPLETED
SELECT DATE_KEY, CHGOFF_YEAR, DISBURSEMENT_YEAR
FROM DATE_DIMN1
WHERE DISBURSEMENT_YEAR > 2000 and CHGOFF_YEAR > 2000
Findings
Recommendations
Rationale
The execution plan of this statement can be improved by creating one or more indices.
Consider running the Access Advisor to improve the physical schema design or creating the recommended index.
Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption.
The execution plan of this statement can be improved by creating one or more indices.
Consider running the Access Advisor to improve the physical schema design or creating the recommended index.
Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption.
Table "SQL926"."DATE_DIMN1" was not analyzed.
Consider collecting optimizer statistics for this table.
The optimizer requires up-to-date statistics for the table in order to select a good execution plan.
Line 181 Column 1
Insert
Modified
Unix/Mac: LF

Part 8: Visualization of the problem statement

Section 8.1 Data Visualization

- The parameters taken are Borrower city and borrower state from the dimension table of location which is mapped with the loan amount approved column of the fact table loan_Fact.
- This visualization helps us understand which state has the highest loan amount approved. i.e., the more loan seekers.
- This can be associated with the jobs generated and retained in those states as the proven source of income must be declared for any loan to be approved.

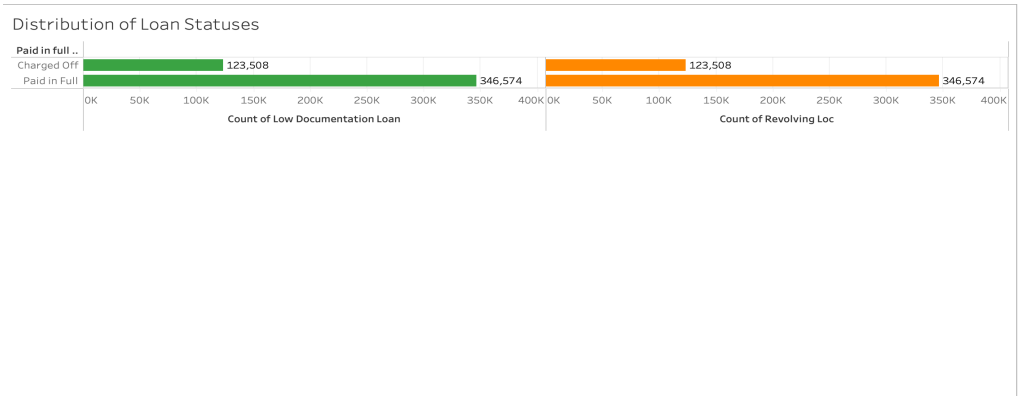


Map based on Longitude (generated) and Latitude (generated). Color shows sum of Loan Approved Bank. Details are shown for State. The view is filtered on State, which excludes Null.

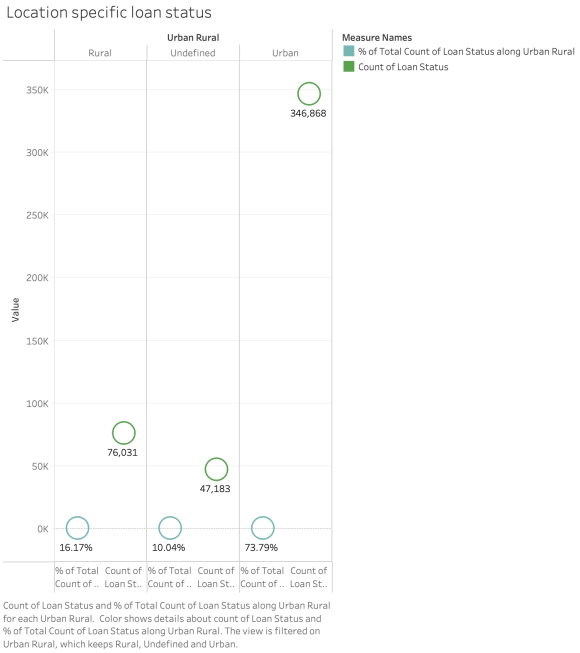
The below visualization portrays businesses with a number of employees of more than 20 within the state of Florida and segmented by each city in the state.

Distribution of loan statuses

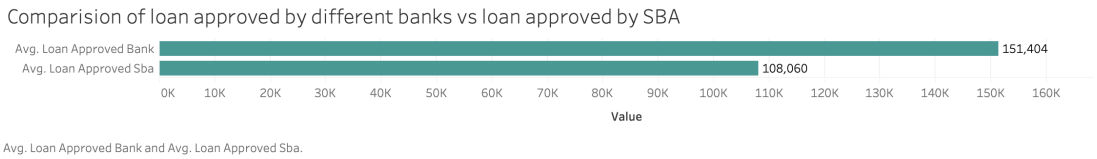
- Charged off loan indicates that the amount at which the loan can be considered defaulted and paid in full loan indicates that the amount has been paid off. We have compared these payments with regard to low documentation loans and revolving lines of credit.



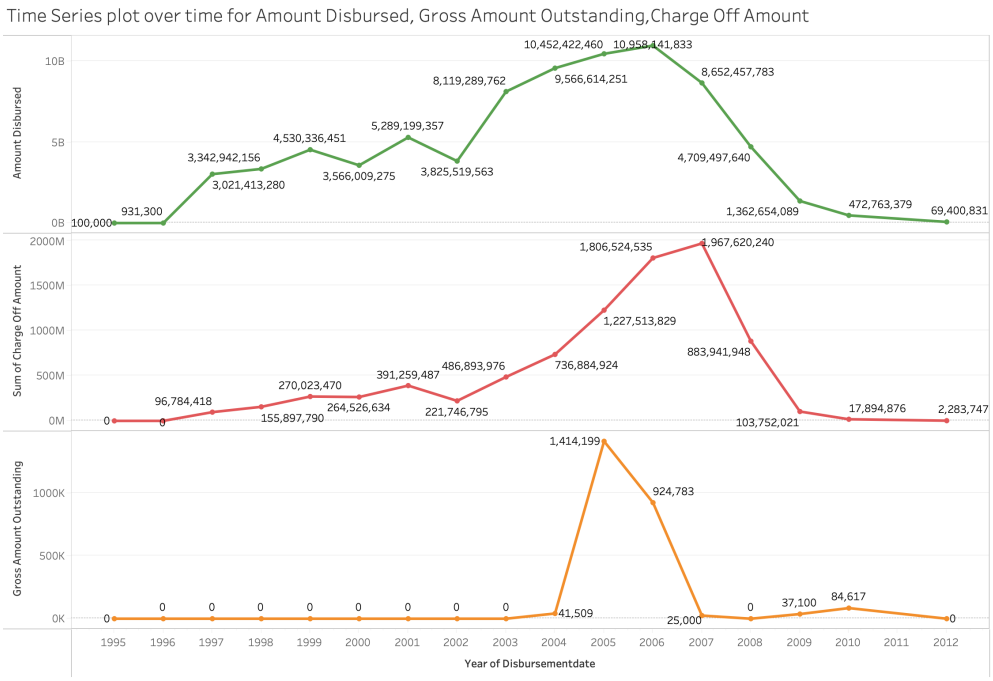
- Based on the urban and rural parameters the count of loans approved was calculated.
- It's observed that the loan applications are more in urban rather than rural areas. Our data didn't have enough parameters to classify the undefined area (we were missing the zip codes, to classify the data), indicating that the urban areas have more job holders compared to that of the rural areas.



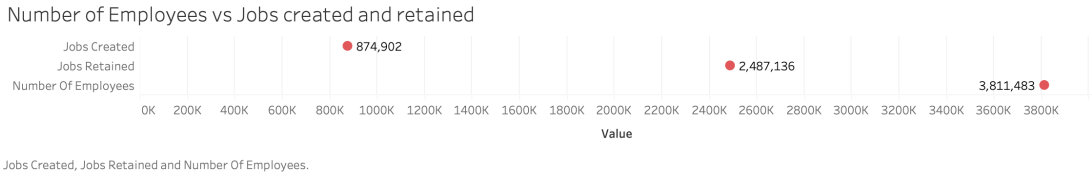
The below visualization illustrates the comparison of bank-approved amounts and SBA-approved amounts for each company in such a manner that the SBA provides an assurance of approved amounts to the bank. It is possible for the lender to easily determine the amount of loan that has been approved through this visualization, and therefore they could decide on opting in for loan either through bank or through SBA.

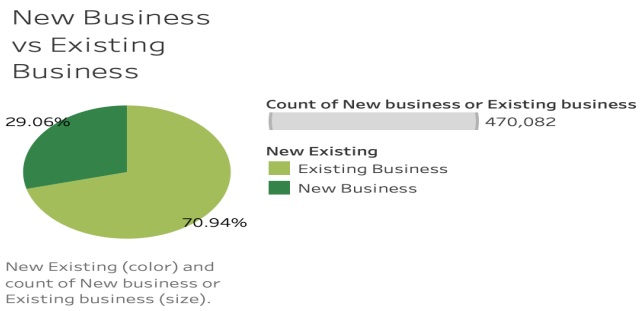


This Visualization helps us understand the number of loans disbursed over the years 1995-2012, the gross outstanding amount, and the charge-off amount. We can understand that the amount disbursed has either been paid off full in most instances i.e., the charge off amount has tallied, due to which the gross amount outstanding has been 0 in most of the cases. We have used the attributes from the date dimension and the loan fact table to achieve this visualization.



The below visualization helps us understand the number of jobs generated by different businesses by taking loans for running their business. The job retention rate is also proportionate to the existing business markets. All the new businesses have contributed to the creation of additional jobs. Despite this the total jobs created and retained are way less than the total available employees for work.





References

- Journal of Statistics Education - "Should This Loan be Approved or Denied?": A Large Dataset with Class Assignment Guidelines - Min Li, Amy Mickel & Stanley Taylor
- The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling by Ralph Kimball (Author), Margy Ross (Author)
- Advance Database Management 2023 (Spring Semester) – Final Report by group 4