

FINAL PROJECT REPORT

ISM 6218 – Advanced Database Management

Submitted by Group 4

Uma Srikanth Reddy Koduru	U94125452
Prasanna Eddala	U24108900
Mohan Vamsi Karasala	U52431177
Akhilasri Musikara	U82181510

Major in
BUSINESS ANALYTICS AND INFORMATION SYSTEMS
Under the guidance of
DR. DONALD BERNDT



MUMA COLLEGE OF BUSINESS
UNIVERSITY OF SOUTH FLORIDA

TOPIC AREA	DESCRIPTION	POINTS	TEAM MEMBERS
Database Design	This part should include a logical database design (for the relational model), using normalization to control redundancy and integrity constraints for data quality.	25	Srikanth Prasanna Mohan Akhila
Query Writing	This part is another chance to write SQL queries, explore transactions, and even do some database programming for stored procedures.	25	Srikanth Prasanna Mohan Akhila
Performance Tuning	In this section, you can capitalize and extend your prior experiments with indexing, optimizer modes, partitioning, parallel execution and any other techniques you want to further explore	25	Srikanth Prasanna Mohan Akhila
Other Topics	Here you are free to explore any other topics of interest. Suggestions include DBA scripts, database security, interface design, data visualization, data mining, and NoSQL databases.	25	Srikanth Prasanna Mohan Akhila

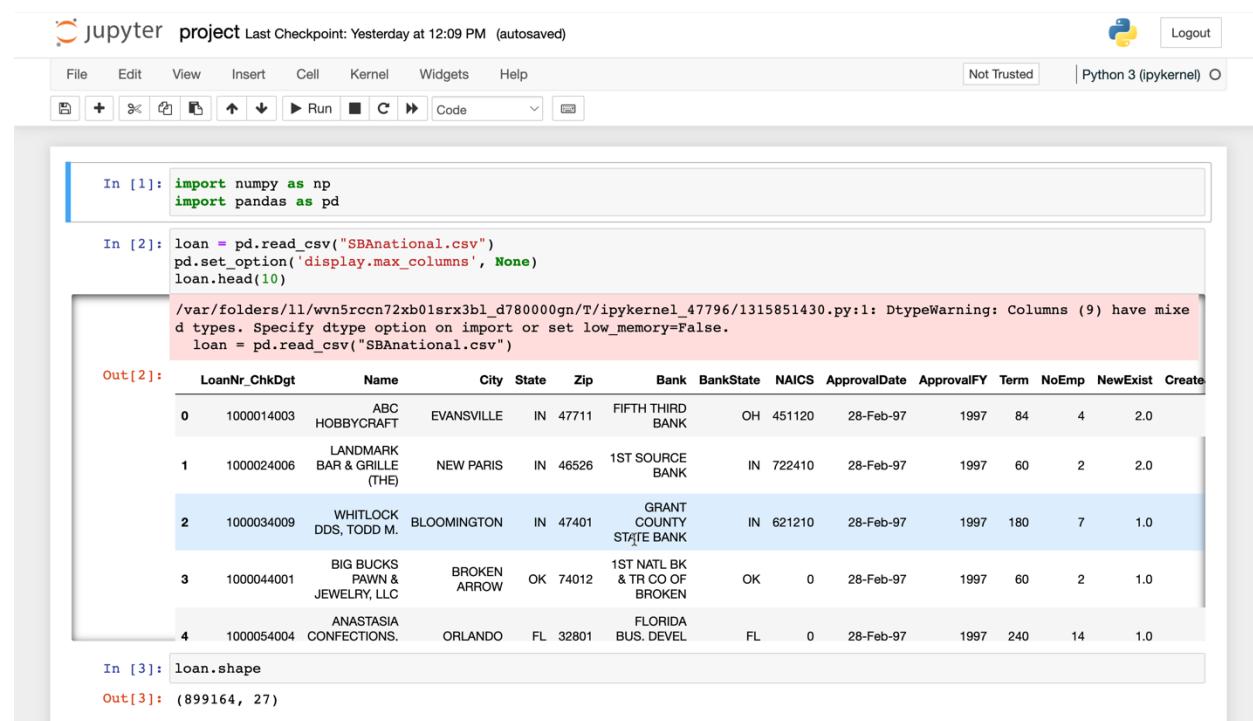
Table of Contents

Introduction	3
Part 1: Database Design.....	5
Section 1.1 Data Integrity.....	6
Section 1.2 Data Generation and Loading.....	8
Part 2: Query Writing.....	10
Section 2.1 Database Programming.....	10
Part 3: Performance Tuning.....	11
Tuning 1.....	11
Tuning 2.....	12
Part 4: Other Topics	14
Data Visualization.....	14
Lessons learnt	16
References	16

Introduction

The data set is about US Small Business Administration and about the banks that have lend to small businesses over a period. A database has been created using this database and visualizations are done to present it to the end user. The technologies used in the project are Python, Oracle SQL Developer, Excel, and Tableau.

First the data is loaded and cleaned using python.



The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** jupyter project Last Checkpoint: Yesterday at 12:09 PM (autosaved)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Not Trusted, Python 3 (ipykernel)
- In [1]:** `import numpy as np`
`import pandas as pd`
- In [2]:** `loan = pd.read_csv("SBANational.csv")`
`pd.set_option('display.max_columns', None)`
`loan.head(10)`
Output: A warning message in red: `/var/folders/11/wvn5rccn72xb01sr3b1_d78000gn/T/ipykernel_47796/1315851430.py:1: DtypeWarning: Columns (9) have mixed types. Specify dtype option on import or set low_memory=False.`
The output table shows 10 rows of data with columns: LoanNr_ChkDgt, Name, City, State, Zip, Bank, BankState, NAICS, ApprovalDate, ApprovalFY, Term, NoEmp, NewExist, Create.
- Out[2]:** The table data is as follows:

LoanNr_ChkDgt	Name	City	State	Zip	Bank	BankState	NAICS	ApprovalDate	ApprovalFY	Term	NoEmp	NewExist	Create
0	ABC HOBBYCRAFT	EVANSVILLE	IN	47711	FIFTH THIRD BANK	OH	451120	28-Feb-97	1997	84	4	2.0	
1	LANDMARK BAR & GRILLE (THE)	NEW PARIS	IN	46526	1ST SOURCE BANK	IN	722410	28-Feb-97	1997	60	2	2.0	
2	WHITLOCK DDS, TODD M.	BLOOMINGTON	IN	47401	GRANT COUNTY STATE BANK	IN	621210	28-Feb-97	1997	180	7	1.0	
3	BIG BUCKS PAWN & JEWELRY, LLC	BROKEN ARROW	OK	74012	1ST NATL BK & TR CO OF BROKEN	OK	0	28-Feb-97	1997	60	2	1.0	
4	ANASTASIA CONFECTIONS.	ORLANDO	FL	32801	FLORIDA BUS. DEVEL	FL	0	28-Feb-97	1997	240	14	1.0	

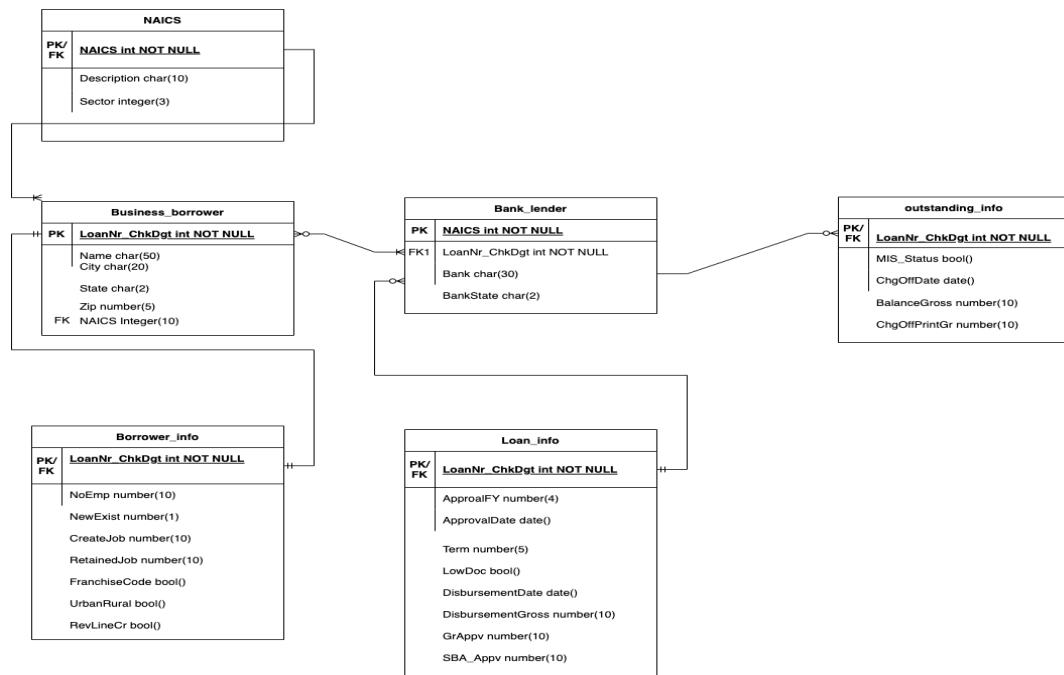
- In [3]:** `loan.shape`
Out[3]: `(899164, 27)`

CODE:

```
import numpy as np
import pandas as pd
loan = pd.read_csv("SBAnational.csv")
pd.set_option('display.max_columns', None)
loan.head(10)
loan.shape
loan.drop_duplicates(inplace = True)
loan.shape
loan = loan[loan['NAICS']!=0]
loan.shape
unique_cols = [col for col in loan.columns if not loan[col].duplicated().any()]
unique_cols
loan['NAICS'].value_counts()
loan.to_csv('loan_data.csv', index=False)
```

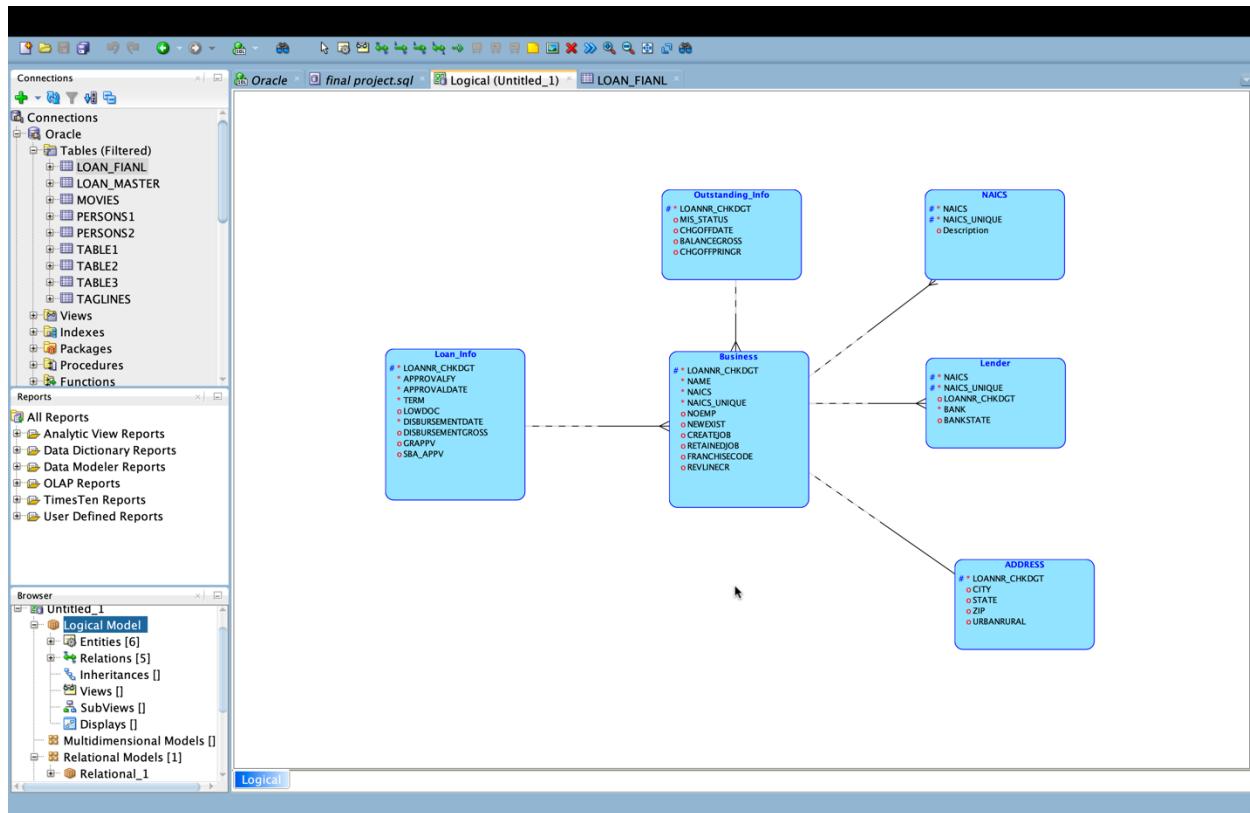
Part 1: Database Design

WITHOUT NORMALIZATION:

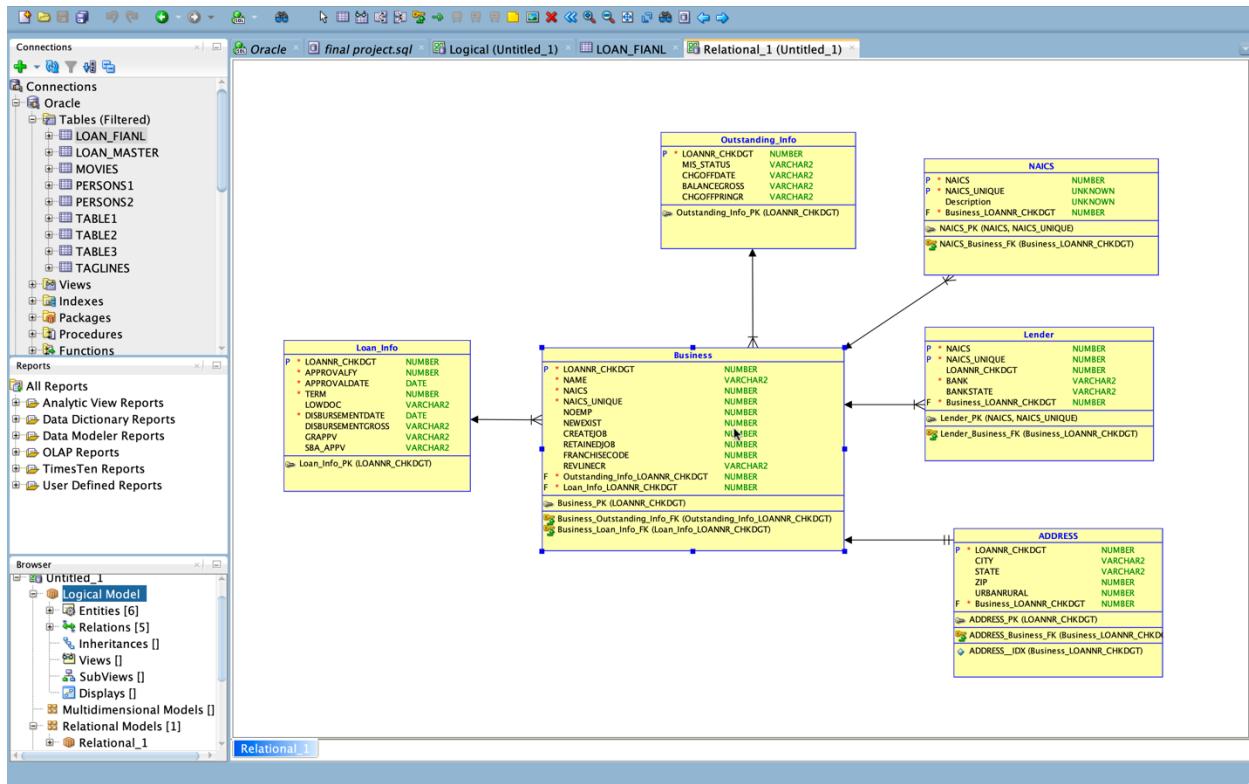


AFTER NORMALIZATION:

The rules of 1NF are followed from the python code, the business_borrower and Borrower_Info tables have been merged to make the design 3NF compliant (No transitive dependencies). The design for the normalized tables is given below. In the below logical diagram all the relations between the tables have been defined and primary keys are set.



The logical diagram is further converted into a relational diagram.



Section 1.1 Data Integrity:

```
ALTER TABLE LOAN_FIANL
```

```
ADD CONSTRAINT naics_as_unique UNIQUE (NAICS_UNIQUE);
```

```
ALTER TABLE BUSINESS
```

```
ADD CONSTRAINT loannr_chkdgt PRIMARY KEY (LOANNR_CHKDG);
```

```
ALTER TABLE IENDER
```

```
ADD CONSTRAINT LENDER_PK PRIMARY KEY (NAICS,NAICS_UNIQUE);
```

```
ALTER TABLE NAICS
```

```
ADD CONSTRAINT NAICS_PK PRIMARY KEY (NAICS,NAICS_UNIQUE);
```

```
ALTER TABLE OUTSTANDING_INFO
```

```
ADD CONSTRAINT OUTSTANDING_PK PRIMARY KEY (LOANNR_CHKDG);
```

ALTER TABLE BUSINESS

```
ADD CONSTRAINT LOAN_INFO_PK PRIMARY KEY (LOANNR_CHKDGT);
```

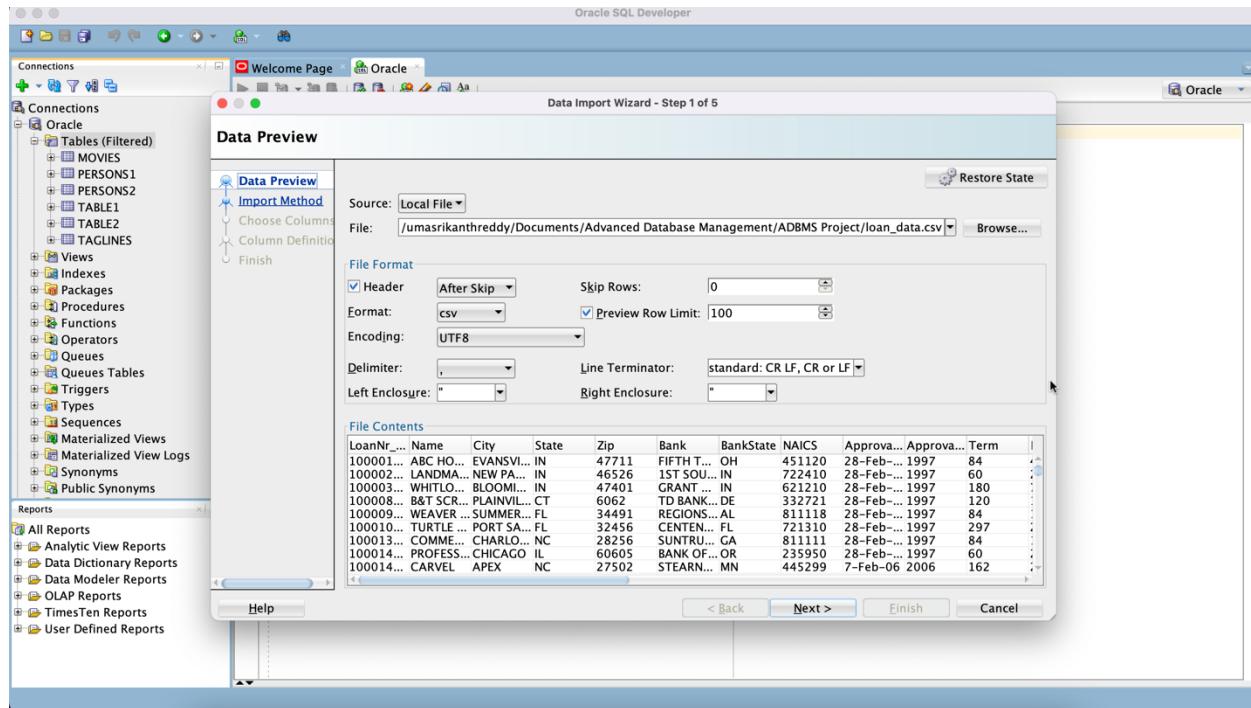
All the primary key constraints have been added to the tables.

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays connections and reports. The main area shows the schema browser with a tree view of relational models, domains, data types, process models, business information, change requests, sensitive types, and TSDP policies. A context menu is open over the 'Relational_1' node, with the 'Add Primary Key' option highlighted. A modal dialog titled 'Add Primary Key' is displayed, prompting for the owner (SQL926), name (LOAN_INFO), and primary key name (LOAN_INFO_PK). The 'Column 1' dropdown is set to LOANNR_CHKDGT. The 'Columns' tab of the LOAN_INFO table is visible in the background, showing nine columns with their respective data types and constraints.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 LOANNR_CHKDGT	NUMBER(38,0)	Yes	(null)	1 (null)	
2 APPROVALDATE	DATE	Yes	(null)	2 (null)	
3 APPROVALFY	NUMBER(38,0)	Yes	(null)	3 (null)	
4 TERM	NUMBER(38,0)	Yes	(null)	4 (null)	
5 LWDODC	VARCHAR2(128 BYTE)	Yes	(null)	5 (null)	
6 DISBURSEMENTDATE	DATE	Yes	(null)	6 (null)	
7 DISBURSEMENTGROSS	VARCHAR2(128 BYTE)	Yes	(null)	7 (null)	
8 GRAPPV	VARCHAR2(128 BYTE)	Yes	(null)	8 (null)	
9 SBA_APPV	VARCHAR2(128 BYTE)	Yes	(null)	9 (null)	

Section 1.2: Data Generation and Loading:

Tables are created using the csv file obtained from the python code.



The NAICS table did not have a unique column in the raw data, so a new column (NAICS_UNIQUE) is created using the code given below. This column together with NAICS column would serve as primary key in the design.

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays connections and reports. The main area has a 'Worksheet' tab selected, showing a query builder window with the following SQL code:

```
CREATE TABLE loan_fianl AS
SELECT
  ROW_NUMBER() OVER (ORDER BY t1.LOANNR_CHKDT) AS NAICS_UNIQUE, t1.*
FROM loan_master t1;
```

Below the worksheet is a 'Script Output' window displaying the message: "Table LOAN_FIANL created." and "Task completed in 4.251 seconds".

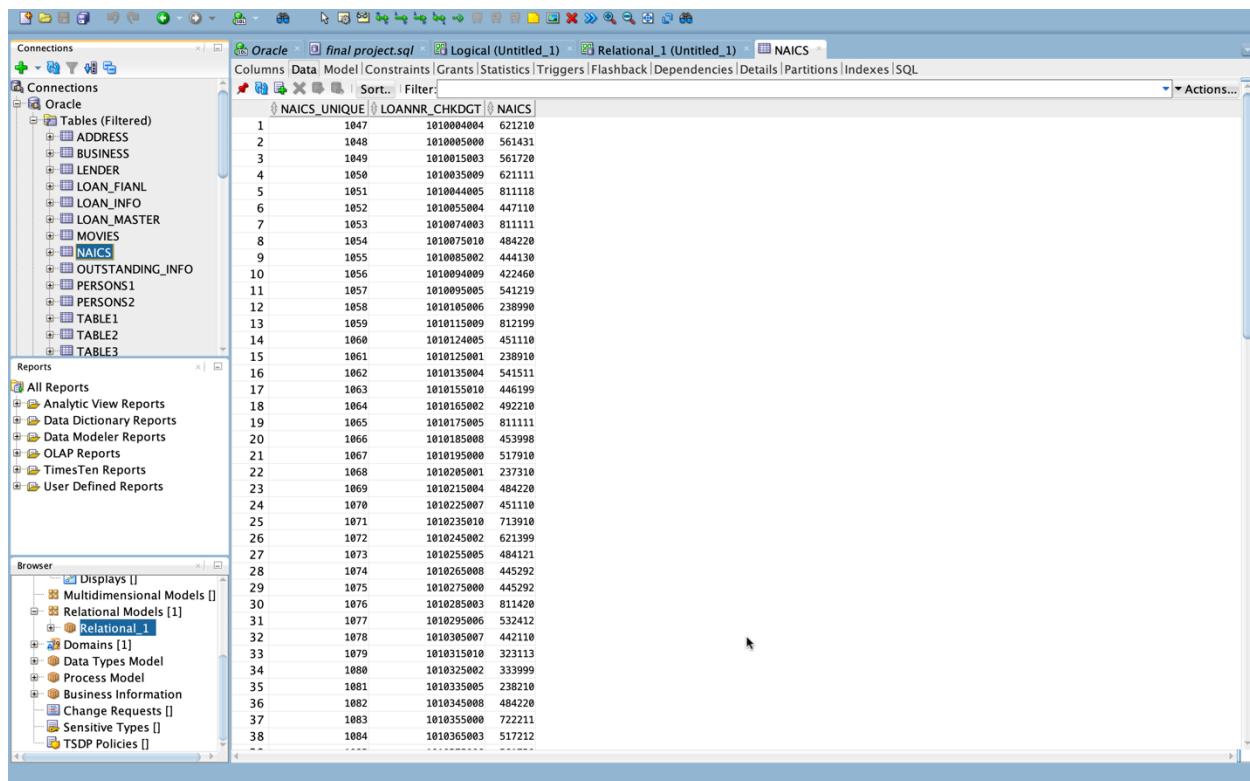
Part 2: Query Writing

Section 2.1: Database Programming:

A stored procedure (create_table) has been created to create a new table with NAICS_UNIQUE column to fill it with unique numbers.

```
CREATE OR REPLACE PROCEDURE create_table AS
BEGIN
  EXECUTE IMMEDIATE 'CREATE TABLE loan_final AS SELECT ROW_NUMBER() OVER (ORDER BY
t1.LOANNR_CHKDG) AS NAICS_UNIQUE, t1.* FROM loan_master t1';
END;
/
EXECUTE create_table;
```

The unique rows are shown in the below picture.



	NAICS_UNIQUE	LOANNR_CHKDG	NAICS
1	1047	1010004004	621210
2	1048	1010005000	561431
3	1049	1010015003	561720
4	1050	1010035009	621111
5	1051	1010044005	811118
6	1052	1010055004	447110
7	1053	1010074003	811111
8	1054	1010075010	484220
9	1055	1010085002	444130
10	1056	1010094009	422460
11	1057	1010095005	541219
12	1058	1010105006	238990
13	1059	1010115009	812199
14	1060	1010124005	451110
15	1061	1010125001	238910
16	1062	1010135004	541511
17	1063	1010155010	446199
18	1064	1010165002	492210
19	1065	1010175005	811111
20	1066	1010185008	453998
21	1067	1010195000	517910
22	1068	1010205001	237310
23	1069	1010215004	484220
24	1070	1010225007	451110
25	1071	1010235010	713910
26	1072	1010245002	621399
27	1073	1010255005	484121
28	1074	1010265008	445292
29	1075	1010275000	445292
30	1076	1010285003	811420
31	1077	1010295006	532412
32	1078	1010305007	442110
33	1079	1010315010	323113
34	1080	1010325002	333999
35	1081	1010335005	238210
36	1082	1010345008	484220
37	1083	1010355000	722211
38	1084	1010365003	517212

Part 3: Performance Tuning

TUNING 1:

- (1) the purpose of the experiment is to change the optimizer mode so as to get in the query results with least computational power.
- (2) A select statement is executed and the cost is noted using the default optimizer mode, then the optimizer mode is set to first 100 rows and the cost is noted.

The screenshot shows the Oracle SQL Developer interface. In the Worksheet tab, there is a SQL script for creating a table and selecting data from three tables: BUSINESS, LENDER, and BANKSTATE. The script includes a procedure to create the table, a constraint to add a primary key, and a select statement to retrieve data where BUSINESS.NOEMP > 20 and LENDER.BANKSTATE = 'FL'. The Explain Plan tab shows the execution plan with various operations like HASH JOIN, NESTED LOOPS, and TABLE ACCESS, along with their costs and cardinalities. The cost for the original query is 61956, while the cost for the tuned query is 1609.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			10015	2910
HASH JOIN	BUSINESS.LOANNR_CHKDG=LENDER.LOANNR_CHKDG		10015	2910
NESTED LOOPS	LENDER	FULL	10015	1301
NESTED LOOPS	STATISTICS COLLECTOR			
TABLE ACCESS	LENDER	FULL	10015	1301
INDEX	LENDER.BANKSTATE='FL'	UNIQUE SCAN		
TABLE ACCESS	BUSINESS	BY INDEX ROWID	1	1609
TABLE ACCESS	BUSINESS	FULL	61956	1609

- (3) There is an improvement in the cost as shown below

The screenshot shows the Oracle SQL Developer interface. The central area is the Worksheet tab, which contains the following SQL code:

```

/*ALTER TABLE BUSINESS
ADD CONSTRAINT loannr_chkdgt PRIMARY KEY (LOANNR_CHKDG);*/
SELECT NAME, NOEMP, BANK, BANKSTATE FROM BUSINESS INNER JOIN LENDER ON
BUSINESS.LOANNR_CHKDG = LENDER.LOANNR_CHKDG
WHERE BUSINESS.NOEMP > 20 AND LENDER.BANKSTATE = 'FL';

/*ALTER SESSION SET OPTIMIZER_MODE = FIRST_ROWS_10;*/

```

Below the worksheet is the Script Output tab, showing the execution time as 0.008 seconds. To the right of the worksheet is the Explain Plan tab, which displays the execution plan with the following details:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			62817	1305
HASH JOIN			62817	1305
Access Predicates	BUSINESS.LOANNR_CHKDG=LENDER.LOANNR_CHKDG			
TABLE ACCESS	LENDER	FULL	10015	1301
Filter Predicates	LENDER.BANKSTATE='FL'			
TABLE ACCESS	BUSINESS	FULL	61956	4
Filter Predicates	BUSINESS.NOEMP>20			

At the bottom of the interface, the status bar indicates "Saved: /Users/umasrikanthreddy/Documents/Advanced Database Management/final project.sql".

TUNING 2:

- (1) The purpose of the below tuning operation is to check whether index has worked or not to reduce the cost.
- (2) A select statement is executed and the cost, Session logical reads are noted, then an index (term_index) is created and the cost are noted.

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle connection selected.
- Worksheet:** Contains the following SQL code:

```

/*ALTER SESSION SET OPTIMIZER_MODE = FIRST_ROWS_10;*/

SELECT BANK, NAME, TERM, DISBURSEMENTDATE FROM LOAN_INFO
INNER JOIN BUSINESS ON LOAN_INFO.LOANNR_CHKDT = BUSINESS.LOANNR_CHKDT
INNER JOIN LENDER ON BUSINESS.LOANNR_CHKDT = LENDER.LOANNR_CHKDT
WHERE TERM >36 AND DISBURSEMENTDATE > '31-JUL-01';

```
- SQL HotSpot:** Shows execution time of 0.859 seconds.
- Autotrace:** Displays the execution plan:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER
SELECT STATEMENT				14285	
HASH JOIN				382350	14285
Access Predicates	BUSINESS.LOANNR_CHKDT=LENDER.LOANNR_CHKDT				
TABLE ACCESS	LENDER	FULL	593479	1300	
HASH JOIN			461869	7676	
Access Predicates					
- Session Logical:** Shows session statistics:

VSSTATNAME	Name	VSMYSTAT	Value
Logical reads		12247	
- Browser:** Relational Models [1] > Relational_1 selected.

(3) There is no change in the cost, but the session logical reads are reduced after creating the index as shown below

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle connection selected.
- Worksheet:** Contains the following SQL code:

```

/*ALTER SESSION SET OPTIMIZER_MODE = FIRST_ROWS_10;*/

SELECT BANK, NAME, TERM, DISBURSEMENTDATE FROM LOAN_INFO
INNER JOIN BUSINESS ON LOAN_INFO.LOANNR_CHKDT = BUSINESS.LOANNR_CHKDT
INNER JOIN LENDER ON BUSINESS.LOANNR_CHKDT = LENDER.LOANNR_CHKDT
WHERE TERM >36 AND DISBURSEMENTDATE > '31-JUL-01';

/*CREATE INDEX term_index ON LOAN_INFO(TERM);*/

```
- SQL HotSpot:** Shows execution time of 0.861 seconds.
- Autotrace:** Displays the execution plan:

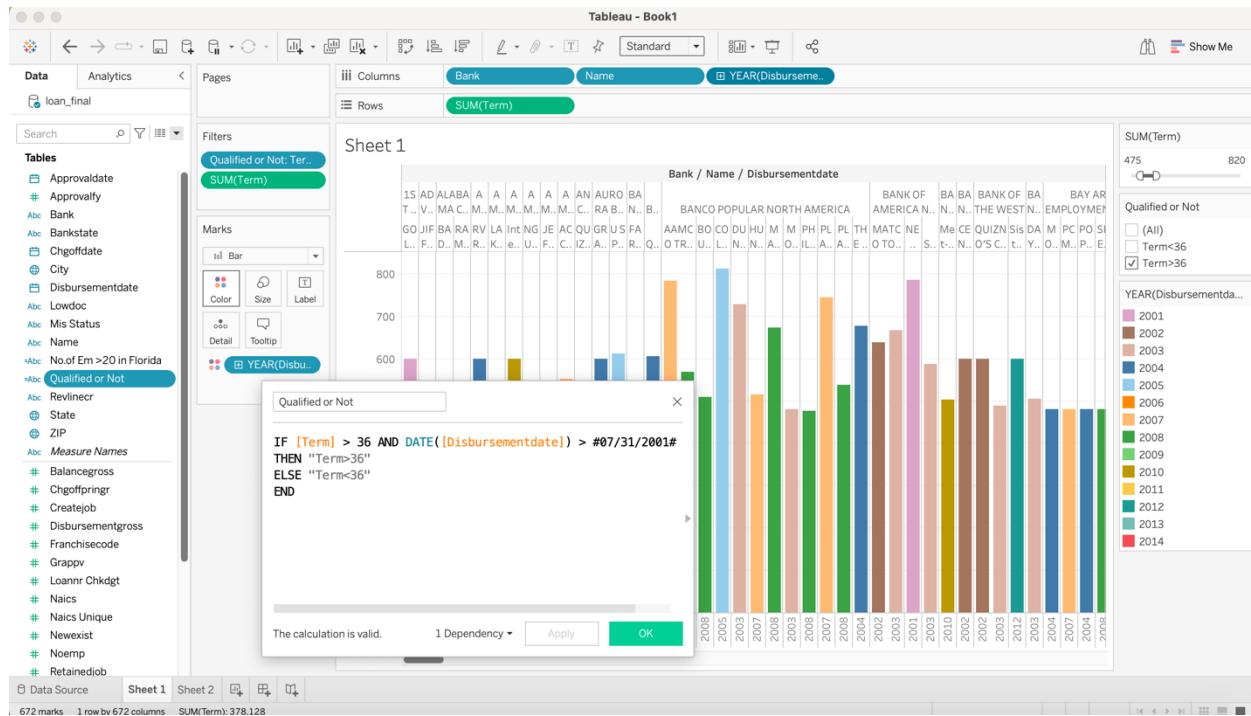
OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER
SELECT STATEMENT				14285	
HASH JOIN				382350	14285
Access Predicates	BUSINESS.LOANNR_CHKDT=LENDER.LOANNR_CHKDT				
TABLE ACCESS	LENDER	FULL	593479	1300	
HASH JOIN			461869	7676	
Access Predicates	LOAN_INFO.LOANNR_CHKDT=BUSINESS.LOANNR_CHKDT				
TABLE ACCESS	LOAN_INFO	FULL	461869	1953	
Filter Predicates					
AND					
- Session Logical:** Shows session statistics:

VSSTATNAME	Name	VSMYSTAT	Value
Logical reads		11933	
- Browser:** Relational Models [1] > Relational_1 selected.

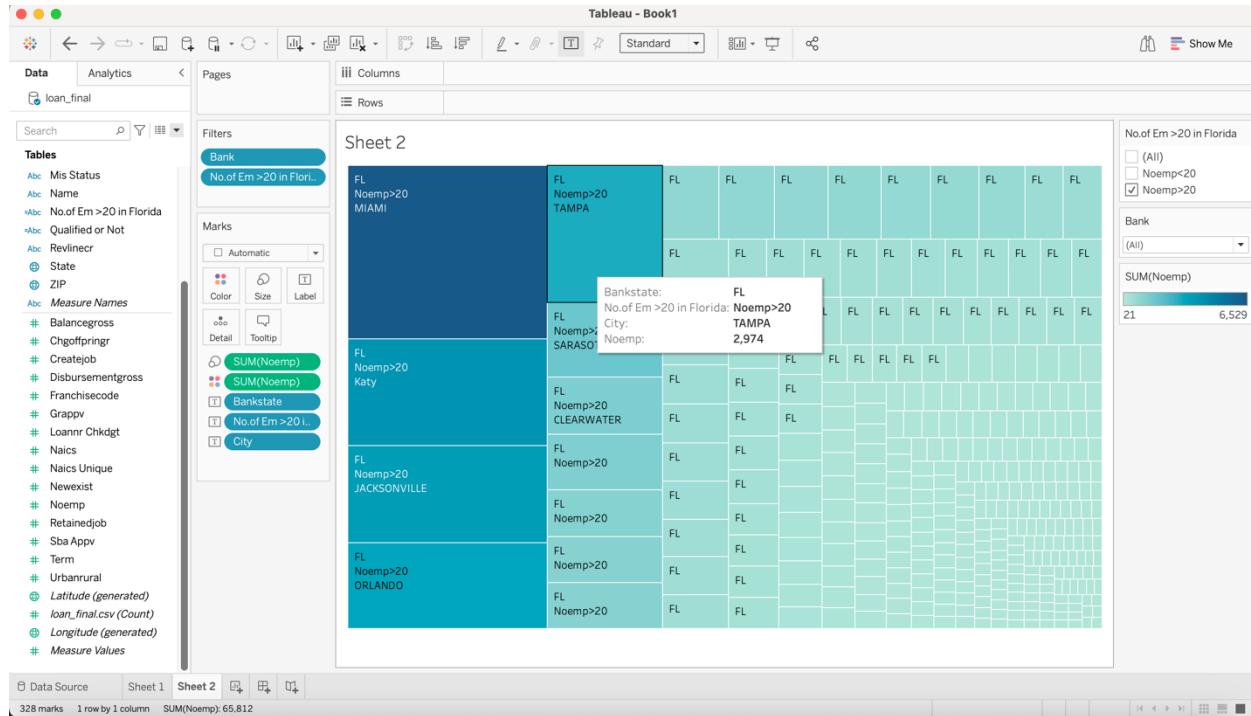
Part 4: Other Topics

Section 4.4: Data Visualization:

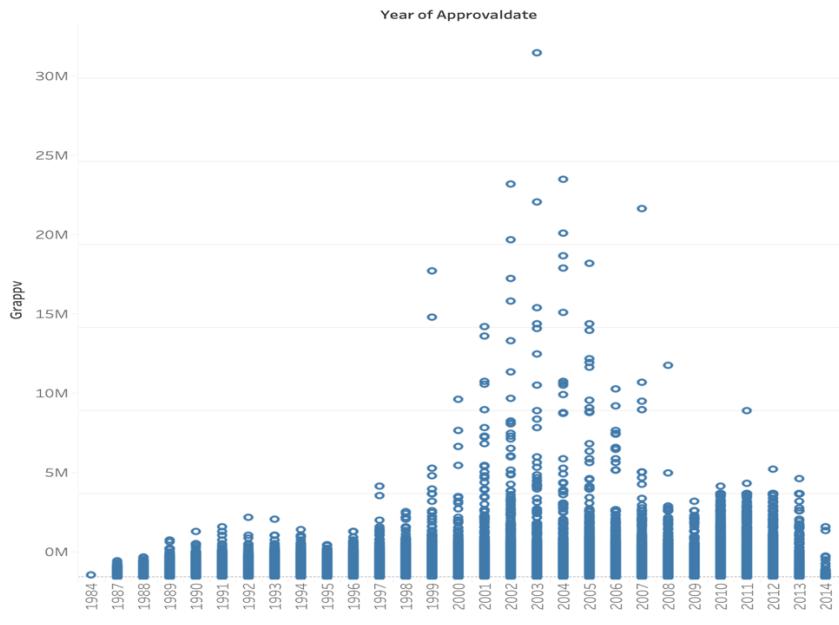
Tableau has been used to visualize data related to the SQL query mentioned in the above sections i.e., the number of businesses who have a loan term greater than 36 months and taken loan after 07/31/2001 are visualized.



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



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, thereby allowing the lender to complete the loan in full. It is possible for the lender to easily determine the amount of funding that has been approved for their company through this visualization.



Lessons learnt

- We have learnt database design and its implementation in a practical way.

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