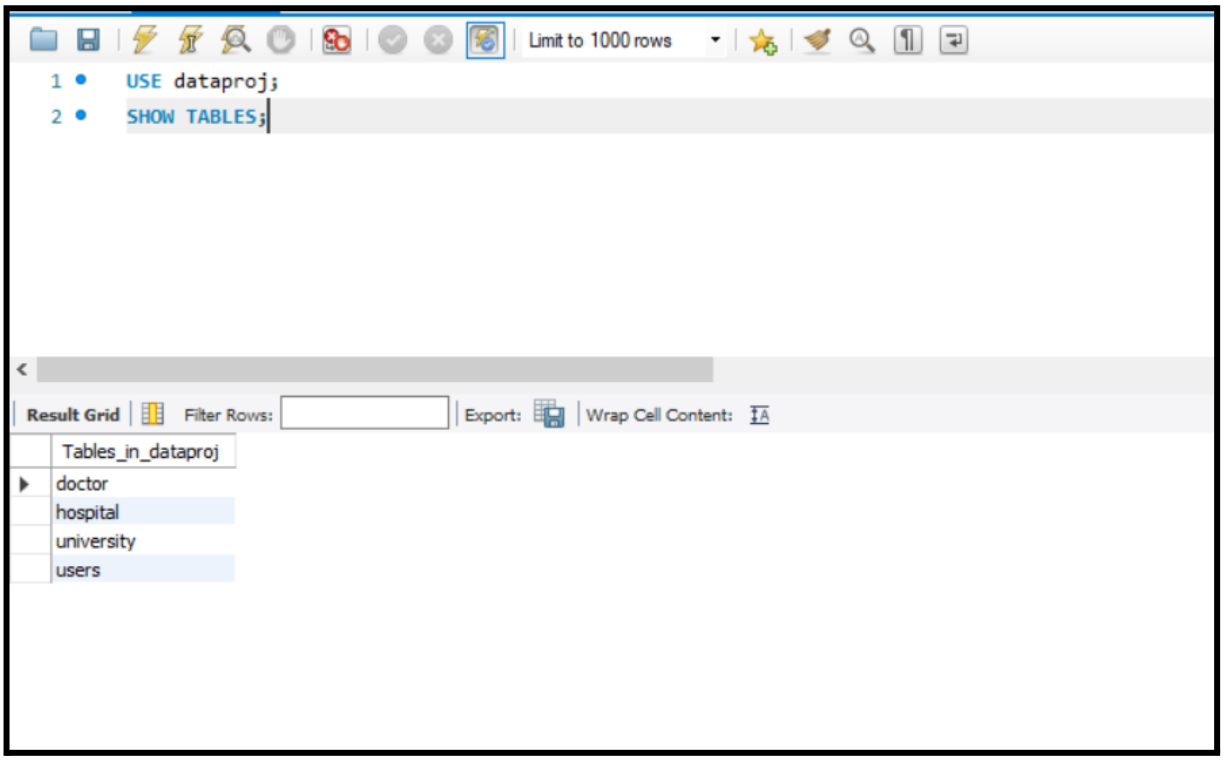
DATABASE DESIGN REPORT

1. The implementation of the four main tables includes:
    a. Users
    b. University
    c. Hospital
    d. Doctor

Tables in Database:
Connection and Schema created.



```
USE dataproj;
SHOW TABLES;
```

```
Tables_in_dataproj
doctor
hospital
university
users
```

2. The DDL commands for the tables are :

Users:
CREATE TABLE Users(User_ID Varchar(255) NOT NULL PRIMARY KEY, `Name` Varchar(255), Address Varchar(255),Phone Varchar(255), Medical History Text(10000), University_ID Varchar(255) NOT NULL,
CONSTRAINT fk_users_university FOREIGN KEY (University_ID)
    REFERENCES University(University_ID)
    ON DELETE CASCADE
);

University:
CREATE TABLE University (University_ID Varchar(255) NOT NULL PRIMARY KEY, `Name` Varchar(255), Telephone Varchar(255), Address Varchar(255)
);

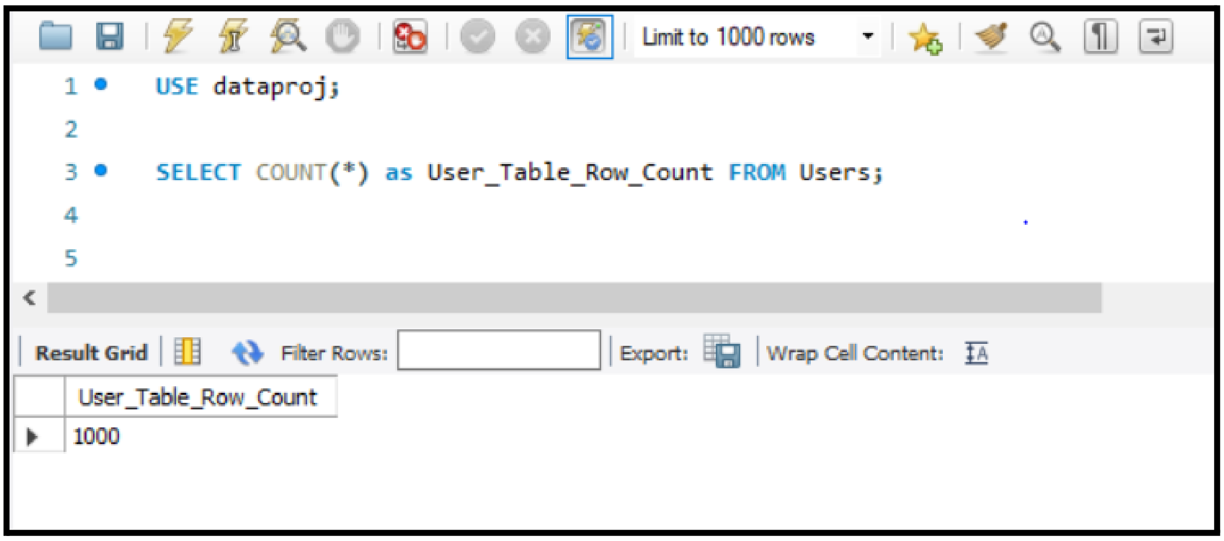
Hospital:
CREATE TABLE Hospital(Hospital_ID Varchar(255) NOT NULL PRIMARY KEY, `Name` Varchar(255), Address Varchar(255), Phone Varchar(255), University_ID Varchar(255) NOT NULL, Distance_From_Univ INT (30), Bed_Availaibility INT(30)
CONSTRAINT fk_hospital_university FOREIGN KEY (University_ID)
REFERENCES University(University_ID)
ON DELETE CASCADE
);

Doctor:
CREATE TABLE Doctor(Doctor_ID Varchar(255) NOT NULL PRIMARY KEY, `First Name` Varchar(255), `Middle Name` Varchar(255). `Last Name` Varchar(255), Address1 Varchar(255), Address2 Varchar(255), City Varchar(255), , State Varchar(255), Zip Varchar(255), Phone Varchar(255), University_ID Varchar(255) NOT NULL,
CONSTRAINT fk_doctor_university FOREIGN KEY (University_ID)
REFERENCES University(University_ID)
ON DELETE CASCADE
);

NOTE: The commands mentioned are specific to mysql workbench. `` is used on the attribute Name as the word Name is a keyword in mysql workbench.

User Table:

Count of rows of the User Table:

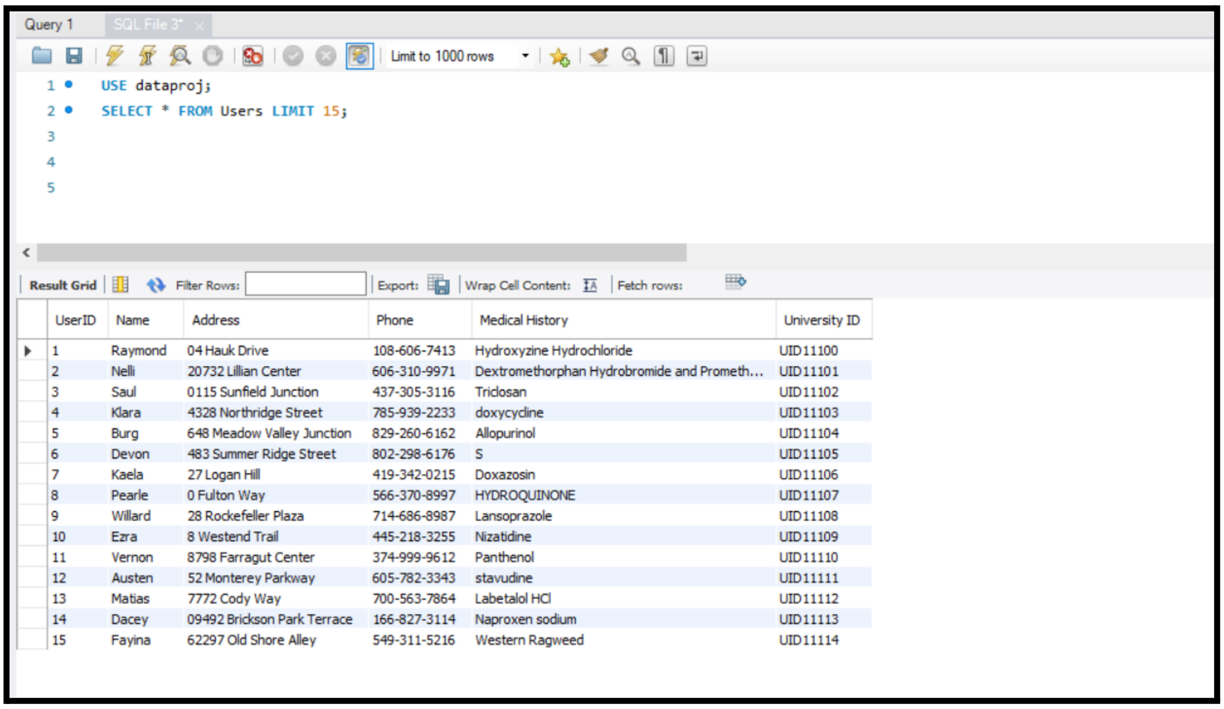


```
USE dataproj;

SELECT COUNT(*) as User_Table_Row_Count FROM Users;
```

User_Table_Row_Count
1000

Top 15 Rows of the Users Table:

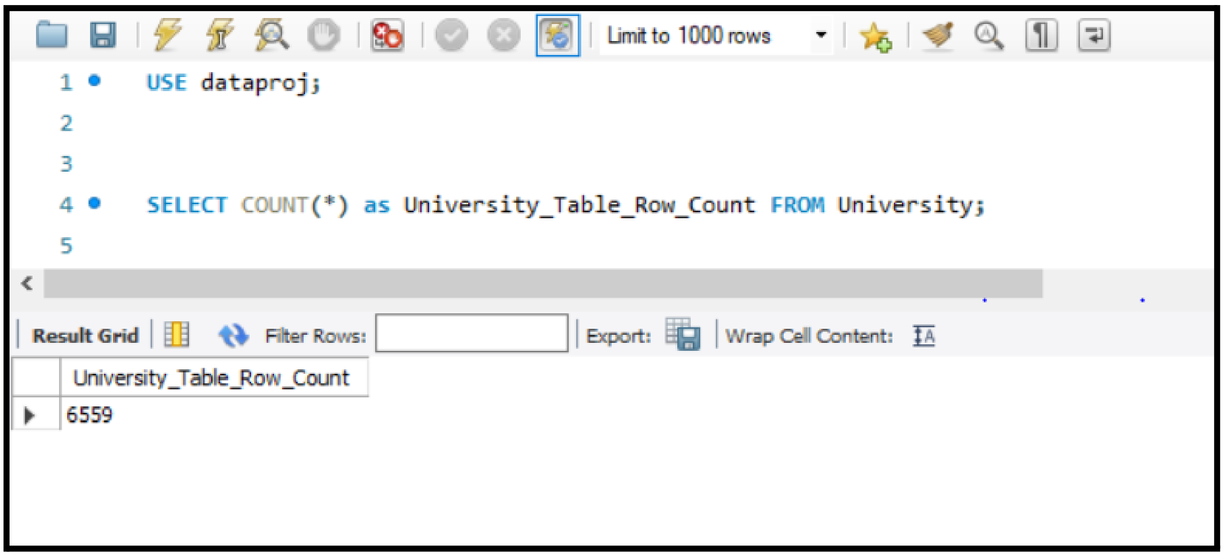


```
USE dataproj;
SELECT * FROM Users LIMIT 15;
```

UserID	Name	Address	Phone	Medical History	University ID
1	Raymond	04 Hauk Drive	108-606-7413	Hydroxyzine Hydrochloride	UID11100
2	Nelli	20732 Lillian Center	606-310-9971	Dextromethorphan Hydrobromide and Prometh...	UID11101
3	Saul	0115 Sunfield Junction	437-305-3116	Triclosan	UID11102
4	Klara	4328 Northridge Street	785-939-2233	doxycycline	UID11103
5	Burg	648 Meadow Valley Junction	829-260-6162	Allopurinol	UID11104
6	Devon	483 Summer Ridge Street	802-298-6176	S	UID11105
7	Kaela	27 Logan Hill	419-342-0215	Doxazosin	UID11106
8	Pearle	0 Fulton Way	566-370-8997	HYDROQUINONE	UID11107
9	Willard	28 Rockefeller Plaza	714-686-8987	Lansoprazole	UID11108
10	Ezra	8 Westend Trail	445-218-3255	Nizatidine	UID11109
11	Vernon	8798 Farragut Center	374-999-9612	Panthenol	UID11110
12	Austen	52 Monterey Parkway	605-782-3343	stavudine	UID11111
13	Matias	7772 Cody Way	700-563-7864	Labetalol HCl	UID11112
14	Dacey	09492 Brickson Park Terrace	166-827-3114	Naproxen sodium	UID11113
15	Fayina	62297 Old Shore Alley	549-311-5216	Western Ragweed	UID11114

__University Table:__

Count of rows of the University Table:

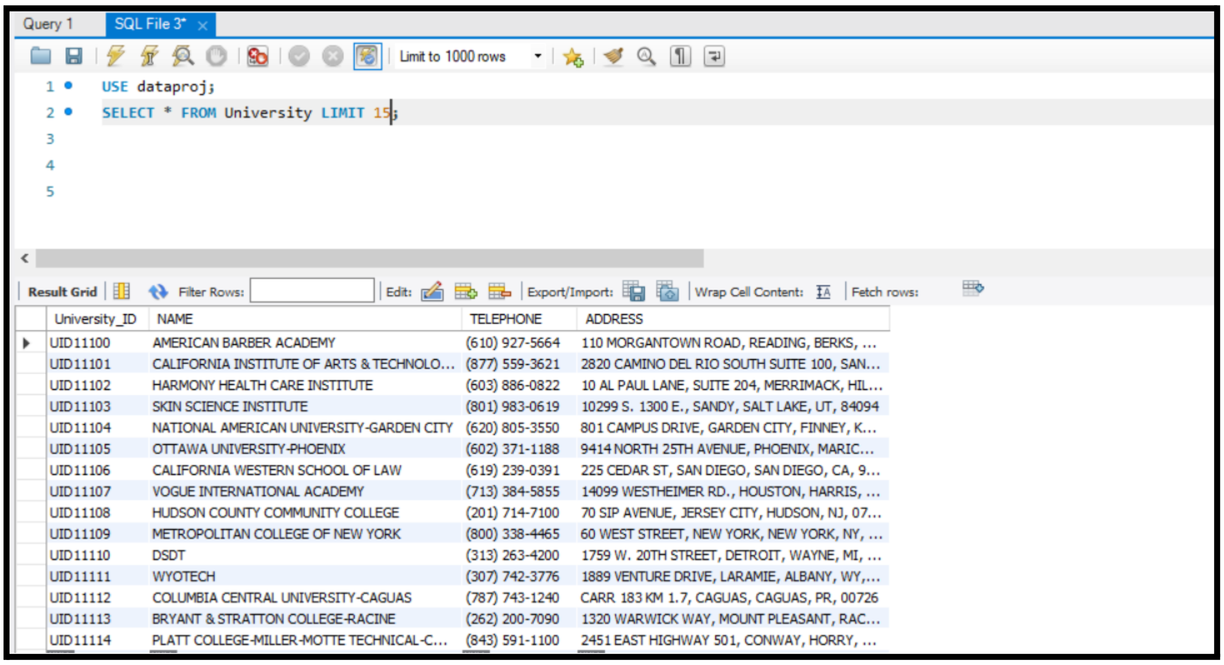


```
USE dataproj;


SELECT COUNT(*) as University_Table_Row_Count FROM University;
```

University_Table_Row_Count
6559

Top 15 Rows of the University Table:

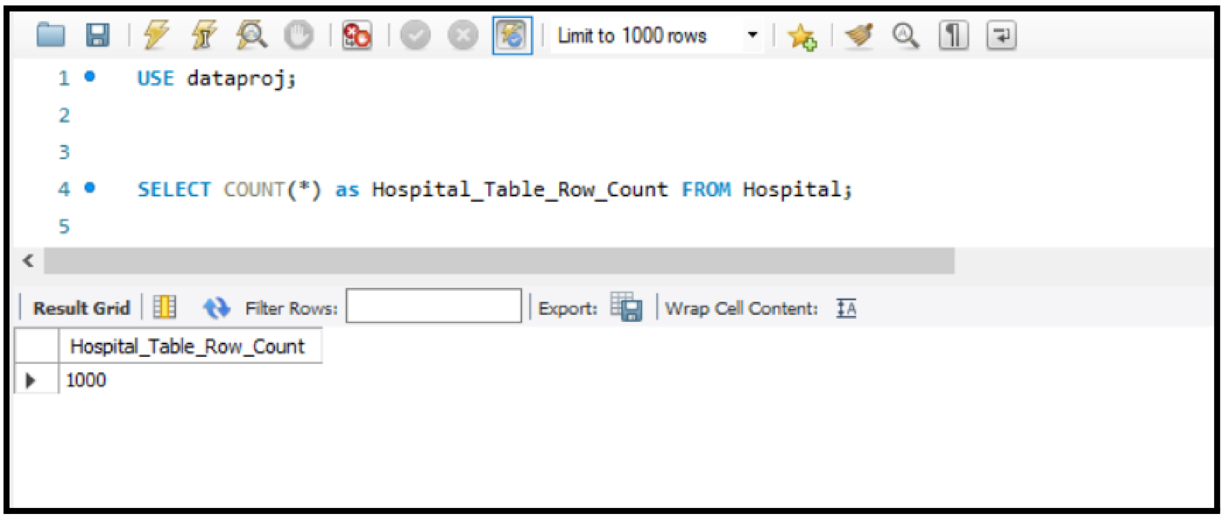


```sql
USE dataproj;
SELECT * FROM University LIMIT 15;
```

University_ID	NAME	TELEPHONE	ADDRESS
UID11100	AMERICAN BARBER ACADEMY	(610) 927-5664	110 MORGANTOWN ROAD, READING, BERKS, ...
UID11101	CALIFORNIA INSTITUTE OF ARTS & TECHNOLO...	(877) 559-3621	2820 CAMINO DEL RIO SOUTH SUITE 100, SAN...
UID11102	HARMONY HEALTH CARE INSTITUTE	(603) 886-0822	10 AL PAUL LANE, SUITE 204, MERRIMACK, HIL...
UID11103	SKIN SCIENCE INSTITUTE	(801) 983-0619	10299 S. 1300 E., SANDY, SALT LAKE, UT, 84094
UID11104	NATIONAL AMERICAN UNIVERSITY-GARDEN CITY	(620) 805-3550	801 CAMPUS DRIVE, GARDEN CITY, FINNEY, K...
UID11105	OTTAWA UNIVERSITY-PHOENIX	(602) 371-1188	9414 NORTH 25TH AVENUE, PHOENIX, MARIC...
UID11106	CALIFORNIA WESTERN SCHOOL OF LAW	(619) 239-0391	225 CEDAR ST, SAN DIEGO, SAN DIEGO, CA, 9...
UID11107	VOGUE INTERNATIONAL ACADEMY	(713) 384-5855	14099 WESTHEIMER RD., HOUSTON, HARRIS, ...
UID11108	HUDSON COUNTY COMMUNITY COLLEGE	(201) 714-7100	70 SIP AVENUE, JERSEY CITY, HUDSON, NJ, 07...
UID11109	METROPOLITAN COLLEGE OF NEW YORK	(800) 338-4465	60 WEST STREET, NEW YORK, NEW YORK, NY, ...
UID11110	DSDT	(313) 263-4200	1759 W. 20TH STREET, DETROIT, WAYNE, MI, ...
UID11111	WYOTECH	(307) 742-3776	1889 VENTURE DRIVE, LARAMIE, ALBANY, WY,...
UID11112	COLUMBIA CENTRAL UNIVERSITY-CAGUAS	(787) 743-1240	CARR 183 KM 1.7, CAGUAS, CAGUAS, PR, 00726
UID11113	BRYANT & STRATTON COLLEGE-RACINE	(262) 200-7090	1320 WARWICK WAY, MOUNT PLEASANT, RAC...
UID11114	PLATT COLLEGE-MILLER-MOTTE TECHNICAL-C...	(843) 591-1100	2451 EAST HIGHWAY 501, CONWAY, HORRY, ...

<u>Hospital Table:</u>

Count of rows of the University Table:

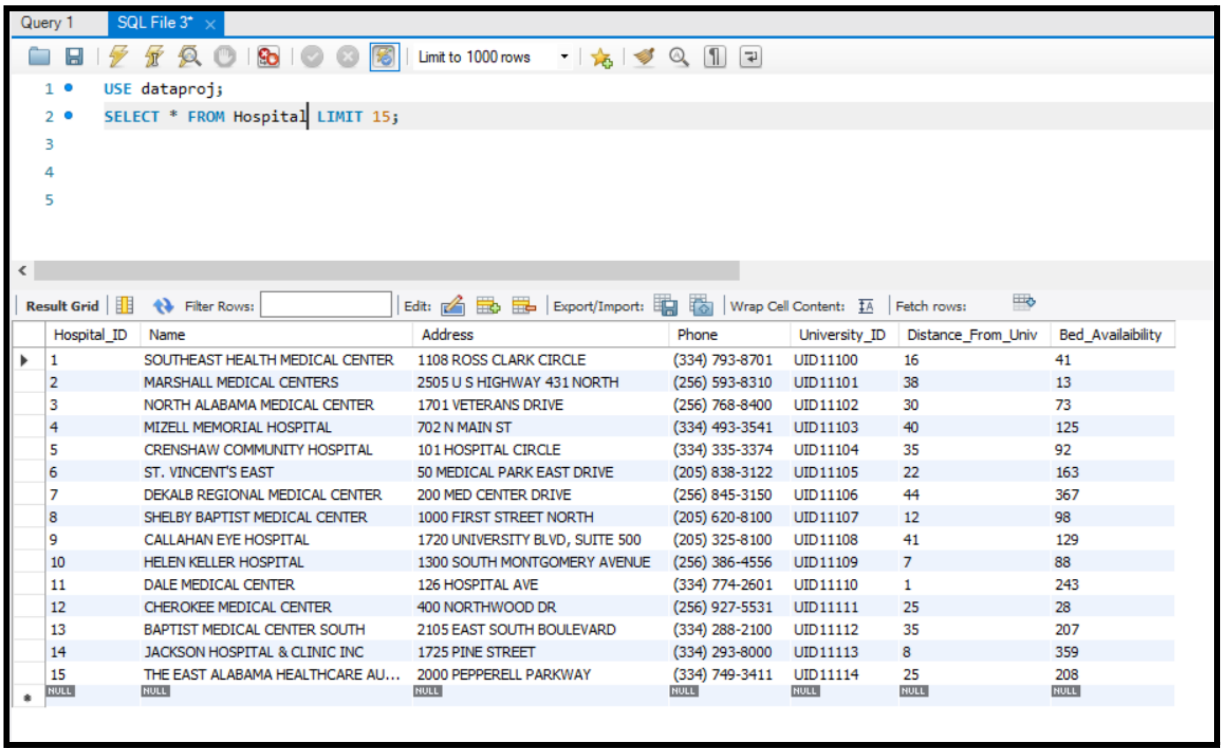


```sql
USE dataproj;


SELECT COUNT(*) as Hospital_Table_Row_Count FROM Hospital;
```

Hospital_Table_Row_Count
1000

Top 15 Rows of the Hospital Table:

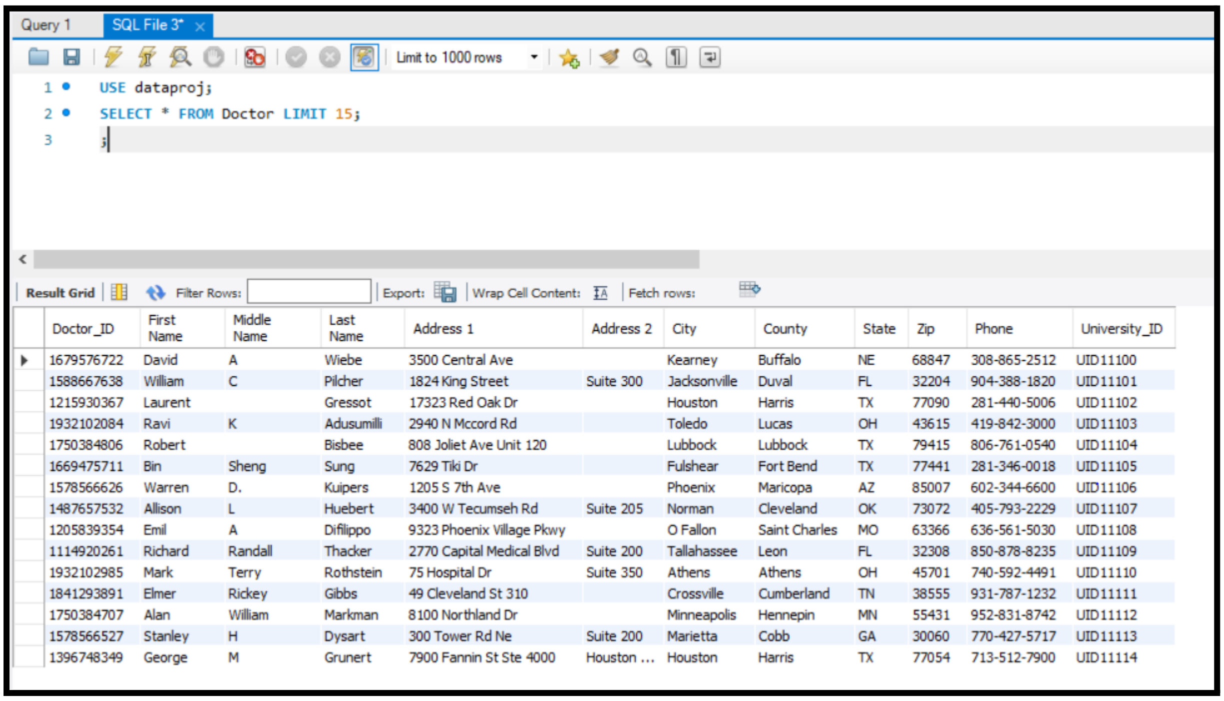


```
USE dataproj;
SELECT * FROM Hospital LIMIT 15;
```

Hospital_ID	Name	Address	Phone	University_ID	Distance_From_Univ	Bed_Availability
1	SOUTHEAST HEALTH MEDICAL CENTER	1108 ROSS CLARK CIRCLE	(334) 793-8701	UID11100	16	41
2	MARSHALL MEDICAL CENTERS	2505 U S HIGHWAY 431 NORTH	(256) 593-8310	UID11101	38	13
3	NORTH ALABAMA MEDICAL CENTER	1701 VETERANS DRIVE	(256) 768-8400	UID11102	30	73
4	MIZELL MEMORIAL HOSPITAL	702 N MAIN ST	(334) 493-3541	UID11103	40	125
5	CRENSHAW COMMUNITY HOSPITAL	101 HOSPITAL CIRCLE	(334) 335-3374	UID11104	35	92
6	ST. VINCENT'S EAST	50 MEDICAL PARK EAST DRIVE	(205) 838-3122	UID11105	22	163
7	DEKALB REGIONAL MEDICAL CENTER	200 MED CENTER DRIVE	(256) 845-3150	UID11106	44	367
8	SHELBY BAPTIST MEDICAL CENTER	1000 FIRST STREET NORTH	(205) 620-8100	UID11107	12	98
9	CALLAHAN EYE HOSPITAL	1720 UNIVERSITY BLVD, SUITE 500	(205) 325-8100	UID11108	41	129
10	HELEN KELLER HOSPITAL	1300 SOUTH MONTGOMERY AVENUE	(256) 386-4556	UID11109	7	88
11	DALE MEDICAL CENTER	126 HOSPITAL AVE	(334) 774-2601	UID11110	1	243
12	CHEROKEE MEDICAL CENTER	400 NORTHWOOD DR	(256) 927-5531	UID11111	25	28
13	BAPTIST MEDICAL CENTER SOUTH	2105 EAST SOUTH BOULEVARD	(334) 288-2100	UID11112	35	207
14	JACKSON HOSPITAL & CLINIC INC	1725 PINE STREET	(334) 293-8000	UID11113	8	359
15	THE EAST ALABAMA HEALTHCARE AU...	2000 PEPPERELL PARKWAY	(334) 749-3411	UID11114	25	208
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Doctor Table:

Top 15 Rows of the Doctor Table:



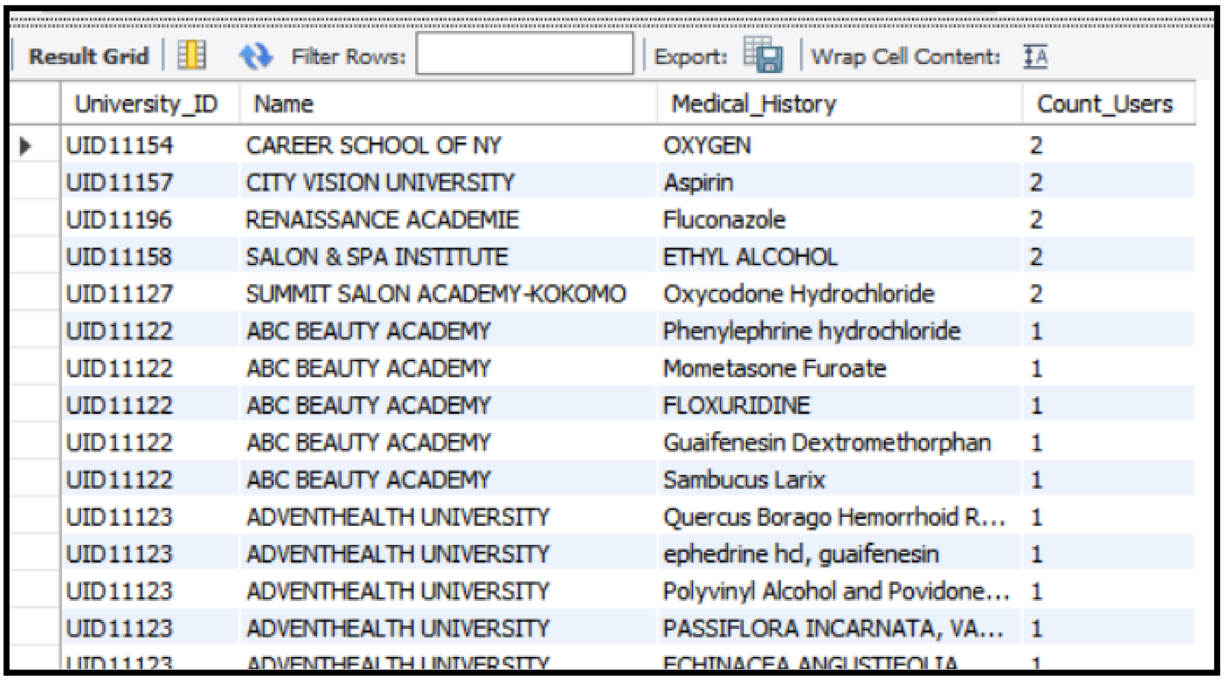
```
USE dataproj;
SELECT * FROM Doctor LIMIT 15;
;
```

Doctor_ID	First Name	Middle Name	Last Name	Address 1	Address 2	City	County	State	Zip	Phone	University_ID
1679576722	David	A	Wiebe	3500 Central Ave		Kearney	Buffalo	NE	68847	308-865-2512	UID11100
1588667638	William	C	Pilcher	1824 King Street	Suite 300	Jacksonville	Duval	FL	32204	904-388-1820	UID11101
1215930367	Laurent		Gressot	17323 Red Oak Dr		Houston	Harris	TX	77090	281-440-5006	UID11102
1932102084	Ravi	K	Adusumilli	2940 N Mccord Rd		Toledo	Lucas	OH	43615	419-842-3000	UID11103
1750384806	Robert		Bisbee	808 Joliet Ave Unit 120		Lubbock	Lubbock	TX	79415	806-761-0540	UID11104
1669475711	Bin	Sheng	Sung	7629 Tiki Dr		Fulshear	Fort Bend	TX	77441	281-346-0018	UID11105
1578566626	Warren	D.	Kuipers	1205 S 7th Ave		Phoenix	Maricopa	AZ	85007	602-344-6600	UID11106
1487657532	Allison	L	Huebert	3400 W Tecumseh Rd	Suite 205	Norman	Cleveland	OK	73072	405-793-2229	UID11107
1205839354	Emil	A	Difilippo	9323 Phoenix Village Pkwy		O Fallon	Saint Charles	MO	63366	636-561-5030	UID11108
1114920261	Richard	Randall	Thacker	2770 Capital Medical Blvd	Suite 200	Tallahassee	Leon	FL	32308	850-878-8235	UID11109
1932102985	Mark	Terry	Rothstein	75 Hospital Dr	Suite 350	Athens	Athens	OH	45701	740-592-4491	UID11110
1841293891	Elmer	Rickey	Gibbs	49 Cleveland St 310		Crossville	Cumberland	TN	38555	931-787-1232	UID11111
1750384707	Alan	William	Markman	8100 Northland Dr		Minneapolis	Hennepin	MN	55431	952-831-8742	UID11112
1578566527	Stanley	H	Dysart	300 Tower Rd Ne	Suite 200	Marietta	Cobb	GA	30060	770-427-5717	UID11113
1396748349	George	M	Grunert	7900 Fannin St Ste 4000	Houston ...	Houston	Harris	TX	77054	713-512-7900	UID11114

3. Advanced Queries
Query 1:
SELECT uni.University_ID, uni.Name, u.Medical_History, COUNT(*) as Count_Users
FROM users u INNER JOIN Univ uni ON u.University_ID = uni.University_ID
GROUP BY uni.University_ID, u.Medical_History, uni.Name
ORDER BY Count_Users DESC, uni.Name ASC;

Description : The query checks for University Id, University Name, and Users Medical History and Count of the Students having the same Medical History.

The output of the Query is provided below (Minimum 15 rows are shown).



University_ID	Name	Medical_History	Count_Users
UID11154	CAREER SCHOOL OF NY	OXYGEN	2
UID11157	CITY VISION UNIVERSITY	Aspirin	2
UID11196	RENAISSANCE ACADEMIE	Fluconazole	2
UID11158	SALON & SPA INSTITUTE	ETHYL ALCOHOL	2
UID11127	SUMMIT SALON ACADEMY-KOKOMO	Oxycodone Hydrochloride	2
UID11122	ABC BEAUTY ACADEMY	Phenylephrine hydrochloride	1
UID11122	ABC BEAUTY ACADEMY	Mometasone Furoate	1
UID11122	ABC BEAUTY ACADEMY	FLOXURIDINE	1
UID11122	ABC BEAUTY ACADEMY	Guaifenesin Dextromethorphan	1
UID11122	ABC BEAUTY ACADEMY	Sambucus Larix	1
UID11123	ADVENTHEALTH UNIVERSITY	Quercus Borago Hemorrhoid R...	1
UID11123	ADVENTHEALTH UNIVERSITY	ephedrine hcl, guaifenesin	1
UID11123	ADVENTHEALTH UNIVERSITY	Polyvinyl Alcohol and Povidone...	1
UID11123	ADVENTHEALTH UNIVERSITY	PASSIFLORA INCARNATA, VA...	1
UID11123	ADVENTHEALTH UNIVERSITY	ECHINACEA ANGUSTIFOLIA	1

Before indexing:

EXPLAIN ANALYZE SELECT uni.University_ID, uni.Name, u.Medical_History, COUNT(*) as Count_Users
FROM users u INNER JOIN Univ uni ON u.University_ID = uni.University_ID
GROUP BY uni.University_ID, u.Medical_History, uni.Name
ORDER BY Count_Users DESC, uni.Name ASC;

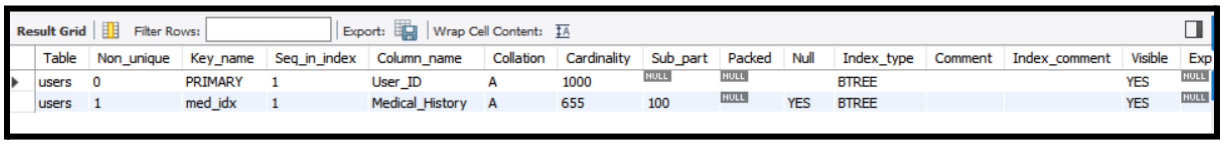
Explain Analyze before indexing:

```
-> Sort: Count_Users DESC, uni.`NAME`  (actual time=62.725..63.149 rows=995 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.082 rows=995 loops=1)
```

```
        -> Aggregate using temporary table  (actual time=15.874..16.070 rows=995 loops=1)
            -> Nested loop inner join  (cost=1210.00 rows=1000) (actual time=5.736..13.464
rows=1000 loops=1)
                -> Filter: (u.University_ID is not null)  (cost=110.00 rows=1000) (actual
time=4.576..9.255 rows=1000 loops=1)
                    -> Table scan on u  (cost=110.00 rows=1000) (actual time=4.574..9.118
rows=1000 loops=1)
                -> Filter: (uni.University_ID = u.University_ID)  (cost=1.00 rows=1) (actual
time=0.004..0.004 rows=1 loops=1000)
                    -> Single-row index lookup on uni using PRIMARY
(University_ID=u.University_ID)  (cost=1.00 rows=1) (actual time=0.003..0.003 rows=1
loops=1000)
```

Adding index on Medical History

Query: CREATE INDEX med_idx ON users (Medical_History(100));



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

	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Exp
▶	users	0	PRIMARY	1	User_ID	A	1000	NULL	NULL		BTREE			YES	NULL
	users	1	med_idx	1	Medical_History	A	655	100	NULL	YES	BTREE			YES	NULL

Explain Analyze After Indexing on Medical History:

```
-> Sort: Count_Users DESC, uni.`NAME`  (actual time=7.333..7.811 rows=995 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.089 rows=995 loops=1)
        -> Aggregate using temporary table  (actual time=6.240..6.437 rows=995 loops=1)
            -> Nested loop inner join  (cost=1209.17 rows=1000) (actual time=0.065..3.665
rows=1000 loops=1)
                -> Filter: (u.University_ID is not null)  (cost=109.17 rows=1000) (actual
time=0.046..1.153 rows=1000 loops=1)
                    -> Table scan on u  (cost=109.17 rows=1000) (actual time=0.046..1.035
rows=1000 loops=1)
                -> Filter: (uni.University_ID = u.University_ID)  (cost=1.00 rows=1) (actual
time=0.002..0.002 rows=1 loops=1000)
                    -> Single-row index lookup on uni using PRIMARY
(University_ID=u.University_ID)  (cost=1.00 rows=1) (actual time=0.002..0.002 rows=1
loops=1000)
```

Dropping index on Medical history

Query: DROP INDEX med_idx on users;

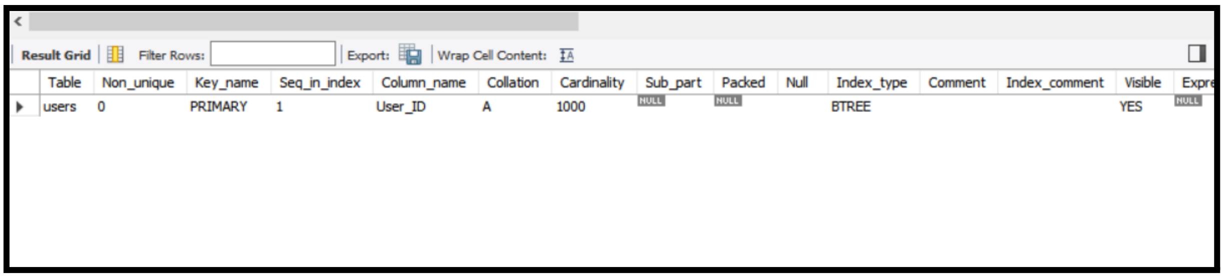


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expre
users	0	PRIMARY	1	User_ID	A	1000	NULL	NULL		BTREE			YES	NULL

Indexing based on university name

Before adding index:

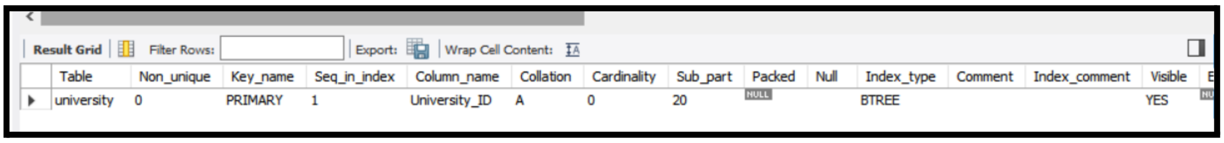


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	E
university	0	PRIMARY	1	University_ID	A	0	20	NULL		BTREE			YES	NU

After adding index using University Name

CREATE INDEX uni_name ON university(name(100));

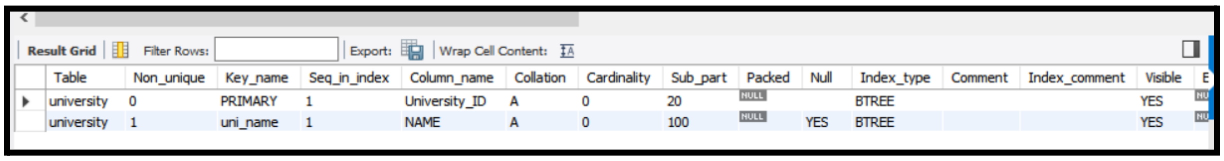


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	E
university	0	PRIMARY	1	University_ID	A	0	20	NULL		BTREE			YES	NU
university	1	uni_name	1	NAME	A	0	100	NULL	YES	BTREE			YES	NU

Explain Analyze after adding Index on University name:

-> Sort: Count_Users DESC, uni.`NAME` (actual time=6.832..7.158 rows=995 loops=1)
    -> Table scan on <temporary> (actual time=0.001..0.095 rows=995 loops=1)
        -> Aggregate using temporary table (actual time=5.895..6.081 rows=995 loops=1)
            -> Nested loop inner join (cost=1209.17 rows=1000) (actual time=0.042..3.694 rows=1000 loops=1)
                -> Filter: (u.University_ID is not null) (cost=109.17 rows=1000) (actual time=0.023..1.164 rows=1000 loops=1)
                    -> Table scan on u (cost=109.17 rows=1000) (actual time=0.023..0.976 rows=1000 loops=1)
                -> Filter: (uni.University_ID = u.University_ID) (cost=1.00 rows=1) (actual time=0.002..0.002 rows=1 loops=1000)
                    -> Single-row index lookup on uni using PRIMARY (University_ID=u.University_ID) (cost=1.00 rows=1) (actual time=0.002..0.002 rows=1 loops=1000)

Explain Analyze after Indexing using both university name and medical history:

```
-> Sort: Count_Users DESC, uni.`NAME`  (actual time=6.792..7.330 rows=995 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.082 rows=995 loops=1)
        -> Aggregate using temporary table  (actual time=5.978..6.181 rows=995 loops=1)
            -> Nested loop inner join  (cost=1209.17 rows=1000) (actual time=0.039..3.785
rows=1000 loops=1)
                -> Filter: (u.University_ID is not null)  (cost=109.17 rows=1000) (actual
time=0.023..1.083 rows=1000 loops=1)
                    -> Table scan on u  (cost=109.17 rows=1000) (actual time=0.022..0.965
rows=1000 loops=1)
                -> Filter: (uni.University_ID = u.University_ID)  (cost=1.00 rows=1) (actual
time=0.002..0.002 rows=1 loops=1000)
                    -> Single-row index lookup on uni using PRIMARY
(University_ID=u.University_ID)  (cost=1.00 rows=1) (actual time=0.002..0.002 rows=1
loops=1000)
```

Analysis on Indexing on Query 1:

Before Indexing: As you can notice above, the aggregate function takes approximately 16 millisecs to complete with 995 rows. Also, the cost on “ table scan on u” is 110 and actual time is approximately 9 millisecs. We can use this as a benchmark to test the indexing results. We consider aggregation time, cost and actual time required on table scans for understanding the analysis.

After adding an index on Medical History: Medical History was picked as an Index because it was a part of the group-by clause which could possibly reduce the cost and actual time. As you can observe, the actual time has reduced to a great extent to 6.4 millisecs. Also, the “table scan on u” actual time was reduced to 1 millisec. The cost however, hasn’t reduced significantly as the number of rows scanned was 1000 before and after indexing on Medical History.

After adding an index on University Name: Similarly, University Name was picked as an Index because the attribute was part of the group-by clause and could potentially reduce the cost and query processing time. As you can observe from above, the aggregate time is 6.1 millisecs and table scan actual time is 0.9 millisecs. These values are quite similar to the indexing values on Medical History with a slight change of 0.1 in aggregate time.

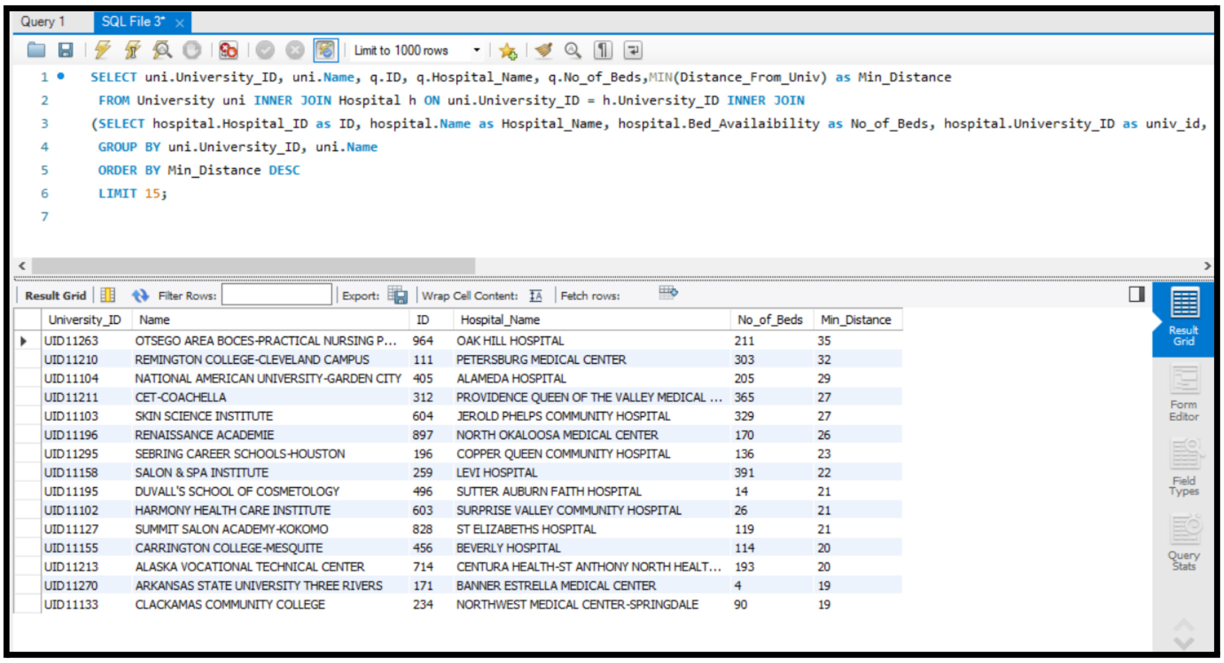
After adding an index on Medical History and University Name: Finally, we applied Indexing using Medical History and University Name together as these attributes are a part of the group-by clause and could potentially reduce the query processing time. As you can notice, the aggregate time is 6 millisecs and table scan time is 0.96 secs which is quite similar to the previous 2 indexing designs.

Hence, picking any one of the above indexing designs would reduce the cost and processing time to a great extent because all the indexing designs produced similar results.

QUERY 2:

SELECT uni.University_ID, uni.Name, q.ID, q.Hospital_Name, q.No_of_Beds,MIN(Distance_From_Univ) as Min_Distance
FROM University uni INNER JOIN Hospital h ON uni.University_ID = h.University_ID INNER JOIN
(SELECT hospital.Hospital_ID as ID, hospital.Name as Hospital_Name, hospital.Bed_Availaibility as No_of_Beds, hospital.University_ID as univ_id, hospital.Distance_From_Univ as dis FROM hospital) q ON (q.univ_id = uni.University_ID AND q.dis = (SELECT MIN(hospital.Distance_From_Univ) FROM hospital WHERE hospital.University_ID = uni.University_ID GROUP BY hospital.University_ID))
GROUP BY uni.University_ID, uni.Name
ORDER BY Min_Distance DESC;

The output of the Query is provided below (Minimum 15 rows are shown).



```
SELECT uni.University_ID, uni.Name, q.ID, q.Hospital_Name, q.No_of_Beds,MIN(Distance_From_Univ) as Min_Distance
 FROM University uni INNER JOIN Hospital h ON uni.University_ID = h.University_ID INNER JOIN
(SELECT hospital.Hospital_ID as ID, hospital.Name as Hospital_Name, hospital.Bed_Availaibility as No_of_Beds, hospital.University_ID as univ_id,
 GROUP BY uni.University_ID, uni.Name
 ORDER BY Min_Distance DESC
 LIMIT 15;
```

University_ID	Name	ID	Hospital_Name	No_of_Beds	Min_Distance
UID11263	OTSEGO AREA BOCES-PRACTICAL NURSING P...	964	OAK HILL HOSPITAL	211	35
UID11210	REMINGTON COLLEGE-CLEVELAND CAMPUS	111	PETERSBURG MEDICAL CENTER	303	32
UID11104	NATIONAL AMERICAN UNIVERSITY-GARDEN CITY	405	ALAMEDA HOSPITAL	205	29
UID11211	CET-COACHELLA	312	PROVIDENCE QUEEN OF THE VALLEY MEDICAL ...	365	27
UID11103	SKIN SCIENCE INSTITUTE	604	JEROLD PHELPS COMMUNITY HOSPITAL	329	27
UID11196	RENAISSANCE ACADEMIE	897	NORTH OKALOOSA MEDICAL CENTER	170	26
UID11295	SEBRING CAREER SCHOOLS-HOUSTON	196	COPPER QUEEN COMMUNITY HOSPITAL	136	23
UID11158	SALON & SPA INSTITUTE	259	LEVI HOSPITAL	391	22
UID11195	DUVALL'S SCHOOL OF COSMETOLOGY	496	SUTTER AUBURN FAITH HOSPITAL	14	21
UID11102	HARMONY HEALTH CARE INSTITUTE	603	SURPRISE VALLEY COMMUNITY HOSPITAL	26	21
UID11127	SUMMIT SALON ACADEMY-KOKOMO	828	ST ELIZABETHS HOSPITAL	119	21
UID11155	CARRINGTON COLLEGE-MESQUITE	456	BEVERLY HOSPITAL	114	20
UID11213	ALASKA VOCATIONAL TECHNICAL CENTER	714	CENTURA HEALTH-ST ANTHONY NORTH HEALT...	193	20
UID11270	ARKANSAS STATE UNIVERSITY THREE RIVERS	171	BANNER ESTRELLA MEDICAL CENTER	4	19
UID11133	CLACKAMAS COMMUNITY COLLEGE	234	NORTHWEST MEDICAL CENTER-SPRINGDALE	90	19

Description: The query outputs the Hospitals, with the number of available beds, nearest to each University.

Explain Analyze before Indexing

```
-> Sort: Min_Distance DESC  (actual time=1076.408..1076.458 rows=200 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.062 rows=200 loops=1)
        -> Aggregate using temporary table  (actual time=1076.203..1076.274 rows=200 loops=1)
```

```
            -> Filter: (hospital.University_ID = h.University_ID)  (cost=101223.90
rows=100000) (actual time=1070.925..1073.900 rows=1040 loops=1)
                -> Inner hash join (<hash>(hospital.University_ID)=<hash>(h.University_ID)),
(hospital.Distance_From_Univ = (select #3))  (cost=101223.90 rows=100000) (actual
time=1070.924..1073.474 rows=1040 loops=1)
                    -> Table scan on hospital  (cost=0.03 rows=1000) (actual time=0.025..1.219
rows=1000 loops=1)
                    -> Hash
                        -> Nested loop inner join  (cost=1207.14 rows=1000) (actual
time=0.101..20.630 rows=1000 loops=1)
                            -> Filter: (h.University_ID is not null)  (cost=107.14 rows=1000) (actual
time=0.060..2.383 rows=1000 loops=1)
                                -> Table scan on h  (cost=107.14 rows=1000) (actual time=0.057..1.903
rows=1000 loops=1)
                            -> Filter: (uni.University_ID = h.University_ID)  (cost=1.00 rows=1)
(actual time=0.017..0.018 rows=1 loops=1000)
                                -> Single-row index lookup on uni using PRIMARY
(University_ID=h.University_ID)  (cost=1.00 rows=1) (actual time=0.017..0.017 rows=1
loops=1000)
```

Adding index on university name:

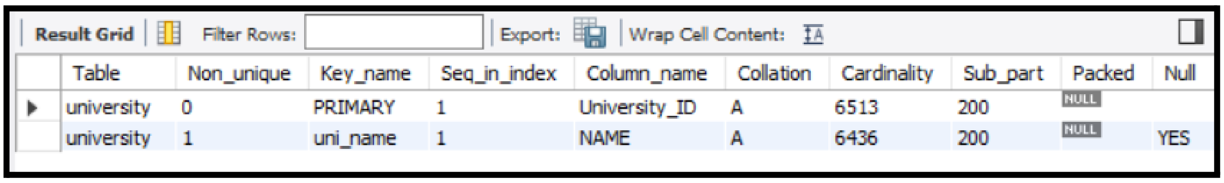
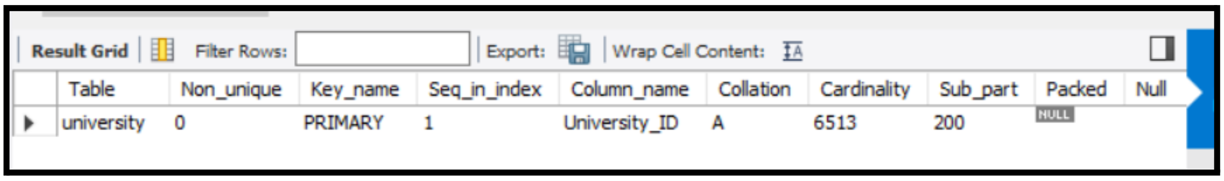


	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null
▶	university	0	PRIMARY	1	University_ID	A	6513	200	NULL	
	university	1	uni_name	1	NAME	A	6436	200	NULL	YES

Explain Analyze after adding index using University Name:

```
-> Sort: Min_Distance DESC  (actual time=1020.550..1020.620 rows=200 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.023 rows=200 loops=1)
        -> Aggregate using temporary table  (actual time=1020.398..1020.439 rows=200
loops=1)
            -> Filter: (hospital.University_ID = h.University_ID)  (cost=101211.19 rows=100000)
(actual time=1016.909..1018.861 rows=1040 loops=1)
                -> Inner hash join (<hash>(hospital.University_ID)=<hash>(h.University_ID)),
(hospital.Distance_From_Univ = (select #3))  (cost=101211.19 rows=100000) (actual
time=1016.908..1018.566 rows=1040 loops=1)
                    -> Table scan on hospital  (cost=0.03 rows=1000) (actual time=0.015..0.891
rows=1000 loops=1)
                    -> Hash
                    -> Nested loop inner join  (cost=1194.43 rows=1000) (actual
time=0.201..10.639 rows=1000 loops=1)
                        -> Filter: (h.University_ID is not null)  (cost=107.14 rows=1000) (actual
time=0.145..2.402 rows=1000 loops=1)
                            -> Table scan on h  (cost=107.14 rows=1000) (actual time=0.142..2.002
rows=1000 loops=1)
                        -> Filter: (uni.University_ID = h.University_ID)  (cost=0.99 rows=1)
(actual time=0.007..0.008 rows=1 loops=1000)
                            -> Single-row index lookup on uni using PRIMARY
(University_ID=h.University_ID)  (cost=0.99 rows=1) (actual time=0.007..0.007 rows=1
loops=1000)
```

Dropping univ name index

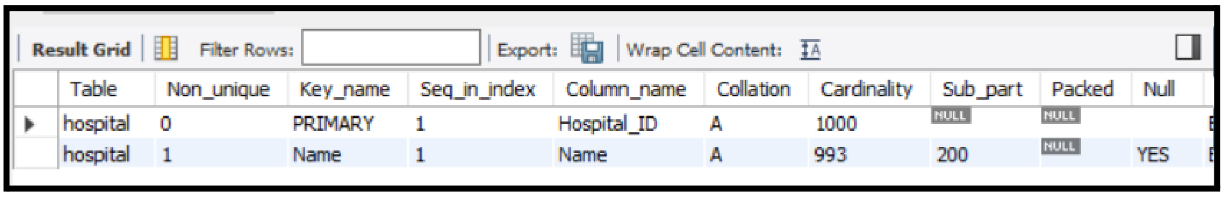


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null
university	0	PRIMARY	1	University_ID	A	6513	200	NULL	

Indexing using Hospital Name

CREATE INDEX Name ON Hospital(Name(200));



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

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null
hospital	0	PRIMARY	1	Hospital_ID	A	1000	NULL	NULL	
hospital	1	Name	1	Name	A	993	200	NULL	YES

Explain Analyze after indexing using Hospital Name:

```
-> Sort: Min_Distance DESC  (actual time=961.378..961.457 rows=200 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.029 rows=200 loops=1)
        -> Aggregate using temporary table  (actual time=961.175..961.247 rows=200 loops=1)
            -> Filter: (hospital.University_ID = h.University_ID)  (cost=101211.19 rows=100000)
(actual time=956.537..959.143 rows=1040 loops=1)
```

```
                -> Inner hash join (<hash>(hospital.University_ID)=<hash>(h.University_ID)),
(hospital.Distance_From_Univ = (select #3))  (cost=101211.19 rows=100000) (actual
time=956.535..958.779 rows=1040 loops=1)
                    -> Table scan on hospital  (cost=0.03 rows=1000) (actual time=0.022..1.113
rows=1000 loops=1)
                    -> Hash
                        -> Nested loop inner join  (cost=1194.43 rows=1000) (actual
time=0.055..10.155 rows=1000 loops=1)
                            -> Filter: (h.University_ID is not null)  (cost=107.14 rows=1000) (actual
time=0.034..2.217 rows=1000 loops=1)
                                -> Table scan on h  (cost=107.14 rows=1000) (actual time=0.033..1.875
rows=1000 loops=1)
                            -> Filter: (uni.University_ID = h.University_ID)  (cost=0.99 rows=1)
(actual time=0.007..0.007 rows=1 loops=1000)
                                -> Single-row index lookup on uni using PRIMARY
(University_ID=h.University_ID)  (cost=0.99 rows=1) (actual time=0.006..0.006 rows=1
loops=1000)
```

Indexing based on both university name and hospital name

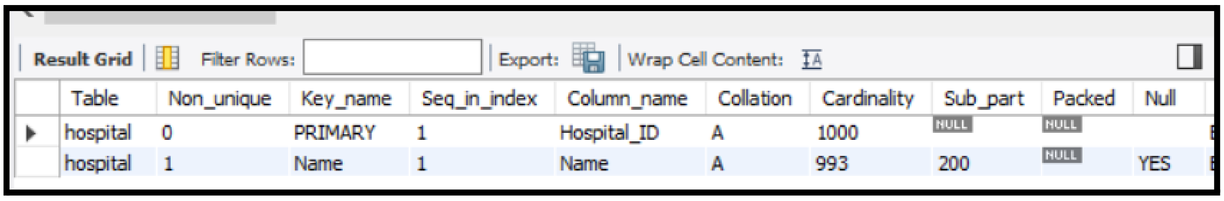


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null
hospital	0	PRIMARY	1	Hospital_ID	A	1000	NULL	NULL	
hospital	1	Name	1	Name	A	993	200	NULL	YES

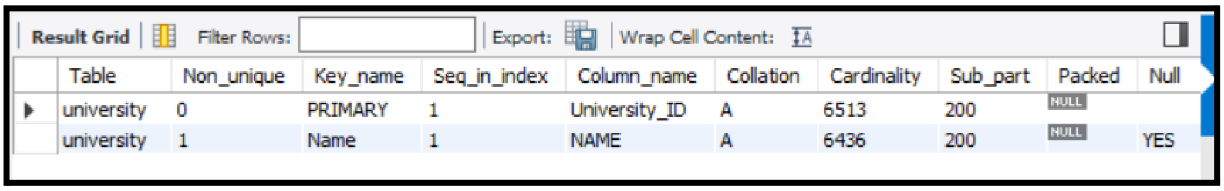


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null
university	0	PRIMARY	1	University_ID	A	6513	200	NULL	
university	1	Name	1	NAME	A	6436	200	NULL	YES

Explain Analyze after indexing using Hospital Name and University Name:

```
-> Sort: Min_Distance DESC  (actual time=827.945..828.054 rows=200 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.032 rows=200 loops=1)
        -> Aggregate using temporary table  (actual time=827.710..827.790 rows=200 loops=1)
            -> Filter: (hospital.University_ID = h.University_ID)  (cost=101223.90
rows=100000) (actual time=822.942..825.766 rows=1040 loops=1)
                -> Inner hash join (<hash>(hospital.University_ID)=<hash>(h.University_ID)),
(hospital.Distance_From_Univ = (select #3))  (cost=101223.90 rows=100000) (actual
time=822.940..825.391 rows=1040 loops=1)
```

```
                -> Table scan on hospital  (cost=0.03 rows=1000) (actual time=0.013..1.182
rows=1000 loops=1)
                -> Hash
                   -> Nested loop inner join  (cost=1207.14 rows=1000) (actual
time=0.046..8.837 rows=1000 loops=1)
                      -> Filter: (h.University_ID is not null)  (cost=107.14 rows=1000) (actual
time=0.025..2.040 rows=1000 loops=1)
                         -> Table scan on h  (cost=107.14 rows=1000) (actual time=0.024..1.697
rows=1000 loops=1)
                      -> Filter: (uni.University_ID = h.University_ID)  (cost=1.00 rows=1)
(actual time=0.006..0.006 rows=1 loops=1000)
                         -> Single-row index lookup on uni using PRIMARY
(University_ID=h.University_ID)  (cost=1.00 rows=1) (actual time=0.005..0.005 rows=1
loops=1000)
```

Why is indexing providing better results?

By incorporating an index on university name and hospital name (individually and simultaneously) we notice that Table Scan on <Table> (Hospital and University) changes to a Lookup search. This is what gives significant improvement in costs and actual times.

Analysis of Indexing on Query-2:

Before Indexing: The aggregate function takes 1.07 secs to complete execution. Also, the actual time for table scan on hospital is 1.2 millisecs and table scan on h is 1.9 millisecs respectively. The cost for table scan on hospital is 0.03 and table scan on h is 107 respectively. We can use this as a benchmark for the indexing results. We consider the aggregation time, actual time for table scan on hospital and actual for table scan on h to see any significant changes.

After adding Indexing using University Name: University Name was picked for Indexing because it was a part of the group-by clause which could possibly reduce the cost and actual time. Indexing is done on Name, which is neither a primary key nor a foreign key. This led to more time taken because indexing is more suited to columns which can be given a meaningful order. The same is true for the aforementioned and following indexing schemes as well where indexing is not done on primary and foreign keys. As you can observe, the actual time for aggregation takes 1.02 secs. Also, the actual time for table scan on hospital is 0.8 millisecs and table scan on h is 2 millisecs respectively. The cost and query processing time did not see much difference after adding an index on University Name and were approximately similar.

After adding Indexing using Hospital Name: Hospital Name was picked for Indexing though it was not a part of the group-by clause. As you can observe above, the actual time for aggregation takes 0.9 secs. The actual time for table scan on hospital is 1.1 millisecs and table scan on h is 1.8 millisecs respectively. There was a minor reduction in time compared to before indexing and Indexing using University Name. The cost remains the same as before indexing and Indexing using University Name. The cost and time was similar to that of Indexing using University Name or before Indexing.

After adding Indexing using Hospital Name and University Name: As you can observe above, the actual time for aggregation takes 0.8 secs. The actual time for table scan on hospital is 1.1 millisecs and table scan on h is 1.6 millisecs respectively. The cost remains the same as before Indexing. This Indexing however, proves to be better in comparison to the remaining Indexing designs and reduces the actual query processing time to some extent.

Hence, using Hospital Name and University Name as the indexing design, we can reduce the cost and query processing time.