Week 1: SQL Basics

Introduction to Databases and SQL:
● What is a database?
● Introduction to Structured Query Language (SQL).
● Overview of popular SQL database management systems (e.g., MySQL, PostgreSQL).

Basic SQL Queries:
● SELECT statement for data retrieval.
● Filtering data using the WHERE clause.
● Sorting data using the ORDER BY clause.
● Limiting results with the LIMIT clause.

Week 1 covers the fundamentals of SQL, starting with an introduction to databases and SQL itself. Here's a simplified breakdown:

What is a database?
A database is like an organized collection of information stored on a computer. It's a structured way to store, manage, and retrieve data.

Introduction to SQL (Structured Query Language):
SQL is a language used to communicate with databases. It helps in performing various tasks like retrieving, updating, and managing data within a database.

Popular Database Management Systems:
Examples include *MySQL* and *PostgreSQL*, which are software used to create and manage databases.

Basic SQL Queries:

SELECT statement: Retrieves specific data from a database.
WHERE clause: Filters data based on specific conditions.
ORDER BY clause: Sorts the retrieved data in a specified order (like alphabetical or numerical).
LIMIT clause: Sets a maximum number of results to be retrieved.

In essence, this week covers the foundational aspects of databases, SQL, and how to retrieve and manipulate data using basic SQL commands like SELECT, WHERE, ORDER BY, and LIMIT.

Data Modification Statements:
● INSERT, UPDATE, DELETE queries to add, update, and delete data.
● Best practices and considerations for data modification.

Data Cleaning and Data Manipulation:

● Identifying and handling missing or incorrect data.
● Using SQL functions for data cleansing (TRIM, LOWER, UPPER, etc.).
● Transforming data using string and date functions.

Here is a simplified breakdown of these concepts:

Data Modification Statements:

INSERT: Adds new data into a database.
UPDATE: Modifies existing data within a database.
DELETE: Removes data from a database.

These statements allow you to add, change, or remove information in a structured manner. Best practices involve being cautious while modifying data, ensuring accuracy, and considering backups before making significant changes.

Data Cleaning and Data Manipulation:

Identifying Missing or Incorrect Data: Recognizing and dealing with data that's either incomplete or inaccurate.
Using SQL Functions for Data Cleansing: Functions like *TRIM* (removes extra spaces), *LOWER* (converts text to lowercase), *UPPER* (converts text to uppercase), etc., help clean and standardize data.
Transforming Data: Utilizing functions specific to strings (text) or dates to change their format or structure within the database.

This part focuses on maintaining data accuracy by recognizing and handling missing or incorrect data, utilizing SQL functions to clean and transform data, and following best practices for modifying data within a database.

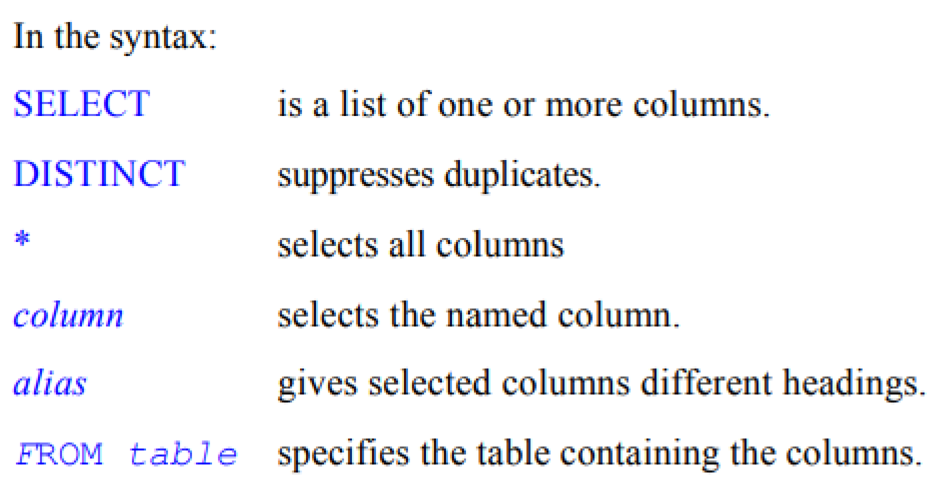
Here are some helpful resources to understand SQL basics, data modification statements, data cleaning, and manipulation:

W3Schools SQL Tutorial: Offers a beginner-friendly introduction to SQL with examples.
https://www.w3schools.com/sql/

Khan Academy - Intro to SQL: Provides a simple introduction to databases and SQL.
https://www.khanacademy.org/computing/computer-programming/sql

SQL Date Functions by SQLTutorial.org: Demonstrates date manipulation functions in SQL.
https://www.sqltutorial.org/sql-date-functions/

PostgreSQL Tutorial by PostgreSQL Tutorial Website: Provides examples and explanations for data modification in PostgreSQL.
https://www.postgresqltutorial.com/



In the syntax:

SELECT	is a list of one or more columns.
DISTINCT	suppresses duplicates.
*	selects all columns
column	selects the named column.
alias	gives selected columns different headings.
FROM *table*	specifies the table containing the columns.

Note: Throughout this module, the words keyword, clause, and statement are used.
A **keyword** refers to an individual SQL element. For example, ***SELECT*** and ***FROM*** are keywords.

A **clause** is a part of an SQL statement. For example. ***SELECT empno, ename,*** … is a clause.

A **statement** is a combination of two or more clauses. For example. ***SELECT * FROM emp is a SQL statement.***

Writing SQL Statements:

Using the following simple rules and guidelines, you can construct valid statements that are both easy to read and easy to edit:

- SQL statements are not case sensitive, unless indicated.
- SQL statements can be entered on one or many lines.
- Keywords cannot be split across lines or abbreviated.
- Clauses are usually placed on separate lines for readability and ease of editing.
- Tabs and indents can be used to make code more readable.
- Keywords typically are entered in uppercase; all other words, such as table names and columns, are entered in lowercase.

Following are some resources which can help you get familiar with MySQL Workbench:
https://www.youtube.com/watch?v=2bW3HuaAUcY&ab_channel=365DataScience

https://youtu.be/x_ez4IlSGOE?feature=shared

Session 1:

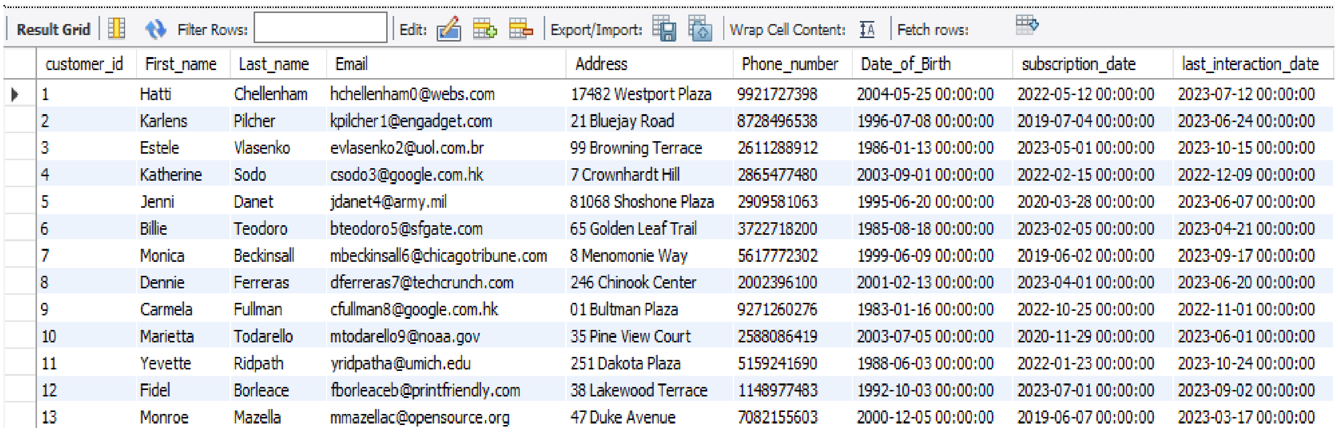
Case Study: Telco Customer Management System

Background:
You’ve been appointed as a Database Analyst for a telecommunications company (“neon telco”) that offers mobile, broadband, and streaming services. They’re looking to revamp their Customer Management System to better track customer, their subscriptions, usage data, and customer service interactions.

Basic SQL Queries:

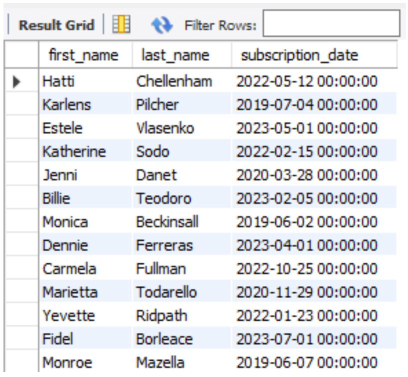
Using SELECT

1. Exercise: Retrieve all columns and rows from the `customer` table.
Query:
*SELECT **
FROM customer;



customer_id	First_name	Last_name	Email	Address	Phone_number	Date_of_Birth	subscription_date	last_interaction_date
1	Hatti	Chellenham	hchellenham0@webs.com	17482 Westport Plaza	9921727398	2004-05-25 00:00:00	2022-05-12 00:00:00	2023-07-12 00:00:00
2	Karlens	Pilcher	kpilcher1@engadget.com	21 Bluejay Road	8728496538	1996-07-08 00:00:00	2019-07-04 00:00:00	2023-06-24 00:00:00
3	Estele	Vlasenko	evlasenko2@uol.com.br	99 Browning Terrace	2611288912	1986-01-13 00:00:00	2023-05-01 00:00:00	2023-10-15 00:00:00
4	Katherine	Sodo	csodo3@google.com.hk	7 Crownhardt Hill	2865477480	2003-09-01 00:00:00	2022-02-15 00:00:00	2022-12-09 00:00:00
5	Jenni	Danet	jdanet4@army.mil	81068 Shoshone Plaza	2909581063	1995-06-20 00:00:00	2020-03-28 00:00:00	2023-06-07 00:00:00
6	Billie	Teodoro	bteodoro5@sfgate.com	65 Golden Leaf Trail	3722718200	1985-08-18 00:00:00	2023-02-05 00:00:00	2023-04-21 00:00:00
7	Monica	Beckinsall	mbeckinsall6@chicagotribune.com	8 Menomonie Way	5617772302	1999-06-09 00:00:00	2019-06-02 00:00:00	2023-09-17 00:00:00
8	Dennie	Ferreras	dferreras7@techcrunch.com	246 Chinook Center	2002396100	2001-02-13 00:00:00	2023-04-01 00:00:00	2023-06-20 00:00:00
9	Carmela	Fullman	cfullman8@google.com.hk	01 Bultman Plaza	9271260276	1983-01-16 00:00:00	2022-10-25 00:00:00	2022-11-01 00:00:00
10	Marietta	Todarello	mtodarello9@noaa.gov	35 Pine View Court	2588086419	2003-07-05 00:00:00	2020-11-29 00:00:00	2023-06-01 00:00:00
11	Yevette	Ridpath	yridpatha@umich.edu	251 Dakota Plaza	5159241690	1988-06-03 00:00:00	2022-01-23 00:00:00	2023-10-24 00:00:00
12	Fidel	Borleace	fborleaceb@printfriendly.com	38 Lakewood Terrace	1148977483	1992-10-03 00:00:00	2023-07-01 00:00:00	2023-09-02 00:00:00
13	Monroe	Mazella	mmazellac@opensource.org	47 Duke Avenue	7082155603	2000-12-05 00:00:00	2019-06-07 00:00:00	2023-03-17 00:00:00

2. Exercise: List only the names and subscription dates of the customer.
Query:
SELECT first_name,last_name,subscription_date
FROM customer;

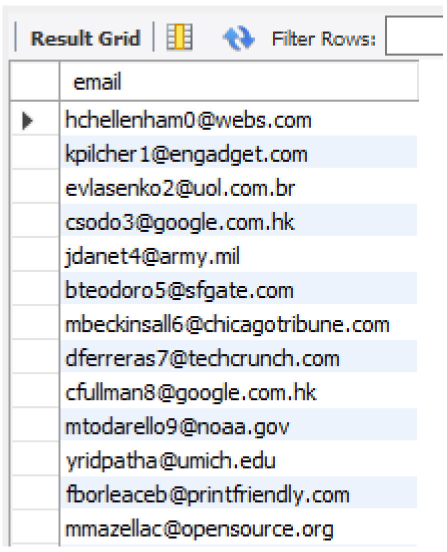


first_name	last_name	subscription_date
Hatti	Chellenham	2022-05-12 00:00:00
Karlens	Pilcher	2019-07-04 00:00:00
Estele	Vlasenko	2023-05-01 00:00:00
Katherine	Sodo	2022-02-15 00:00:00
Jenni	Danet	2020-03-28 00:00:00
Billie	Teodoro	2023-02-05 00:00:00
Monica	Beckinsall	2019-06-02 00:00:00
Dennie	Ferreras	2023-04-01 00:00:00
Carmela	Fullman	2022-10-25 00:00:00
Marietta	Todarello	2020-11-29 00:00:00
Yevette	Ridpath	2022-01-23 00:00:00
Fidel	Borleace	2023-07-01 00:00:00
Monroe	Mazella	2019-06-07 00:00:00

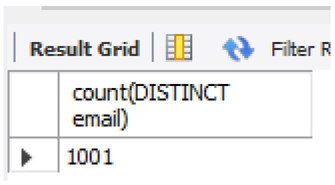
3. Exercise: Fetch all unique email addresses from the `customer` table and then count the number of unique email ids because it should be unique for all.

Query:

SELECT DISTINCT email
FROM customer;
SELECT count(DISTINCT email)
FROM customer;



email
hchellenham0@webs.com
kpilcher1@engadget.com
evlasenko2@uol.com.br
csodo3@google.com.hk
jdanet4@army.mil
bteodoro5@sfgate.com
mbeckinsall6@chicagotribune.com
dferreras7@techcrunch.com
cfullman8@google.com.hk
mtodarello9@noaa.gov
yridpatha@umich.edu
fborleaceb@printfriendly.com
mmazellac@opensource.org

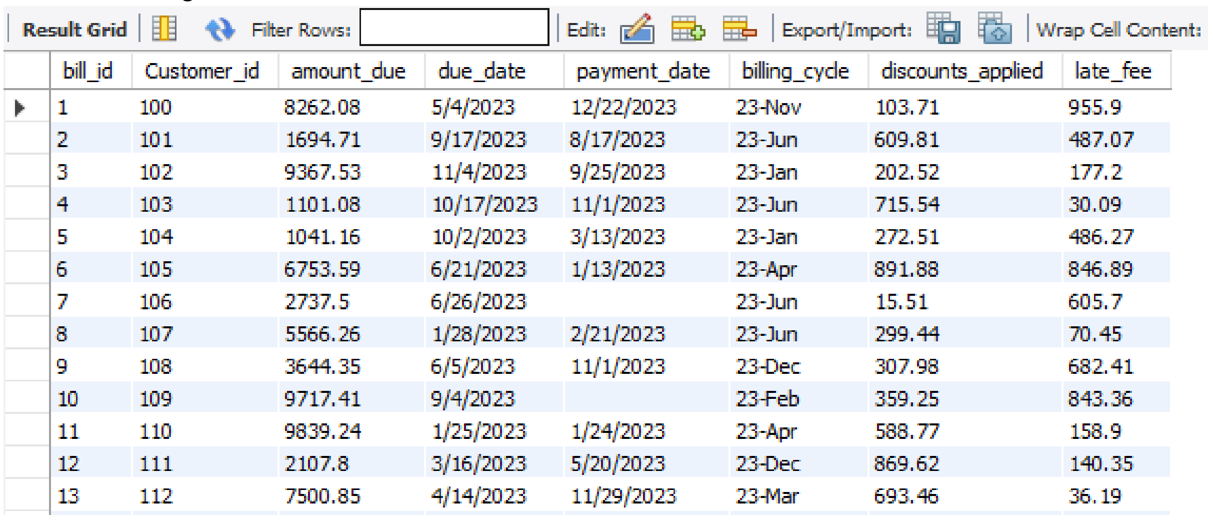


count(DISTINCT email)
1001

4. Exercise: Display all columns from the `billing` table.

Query:

*SELECT **
FROM billing;

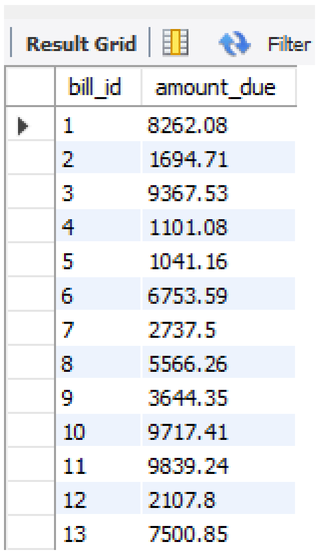


bill_id	Customer_id	amount_due	due_date	payment_date	billing_cycle	discounts_applied	late_fee
1	100	8262.08	5/4/2023	12/22/2023	23-Nov	103.71	955.9
2	101	1694.71	9/17/2023	8/17/2023	23-Jun	609.81	487.07
3	102	9367.53	11/4/2023	9/25/2023	23-Jan	202.52	177.2
4	103	1101.08	10/17/2023	11/1/2023	23-Jun	715.54	30.09
5	104	1041.16	10/2/2023	3/13/2023	23-Jan	272.51	486.27
6	105	6753.59	6/21/2023	1/13/2023	23-Apr	891.88	846.89
7	106	2737.5	6/26/2023		23-Jun	15.51	605.7
8	107	5566.26	1/28/2023	2/21/2023	23-Jun	299.44	70.45
9	108	3644.35	6/5/2023	11/1/2023	23-Dec	307.98	682.41
10	109	9717.41	9/4/2023		23-Feb	359.25	843.36
11	110	9839.24	1/25/2023	1/24/2023	23-Apr	588.77	158.9
12	111	2107.8	3/16/2023	5/20/2023	23-Dec	869.62	140.35
13	112	7500.85	4/14/2023	11/29/2023	23-Mar	693.46	36.19

5. Exercise: Show only the bill ID and the amount due from the `billing` table.

Query:

SELECT bill_id, amount_due
FROM billing;



bill_id	amount_due
1	8262.08
2	1694.71
3	9367.53
4	1101.08
5	1041.16
6	6753.59
7	2737.5
8	5566.26
9	3644.35
10	9717.41
11	9839.24
12	2107.8
13	7500.85

Using WHERE

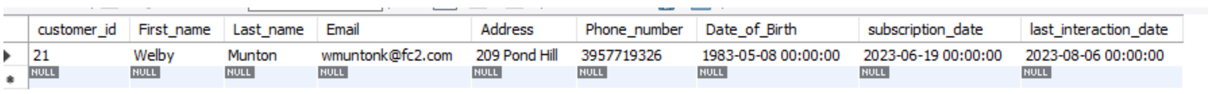
1. Exercise: Identify customer who live at "*209 Pond Hill*".

Query:

*SELECT **

FROM customer

WHERE address = '209 Pond Hill';



customer_id	First_name	Last_name	Email	Address	Phone_number	Date_of_Birth	subscription_date	last_interaction_date
21	Welby	Munton	wmuntonk@fc2.com	209 Pond Hill	3957719326	1983-05-08 00:00:00	2023-06-19 00:00:00	2023-08-06 00:00:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

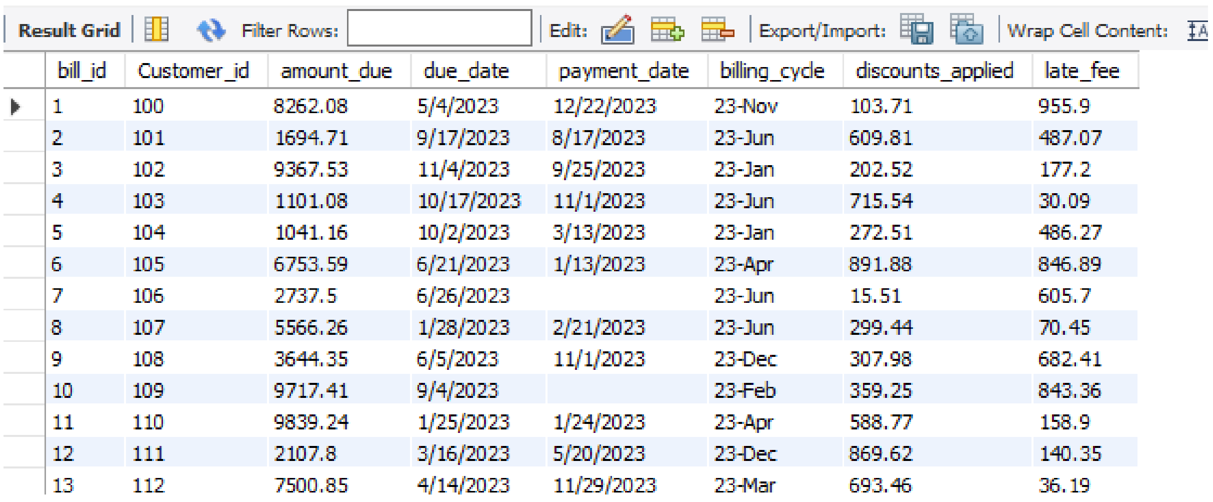
2. Exercise: Find bills in the `billing` table with an amount_due greater than 1000.

Query:

*SELECT **

FROM billing

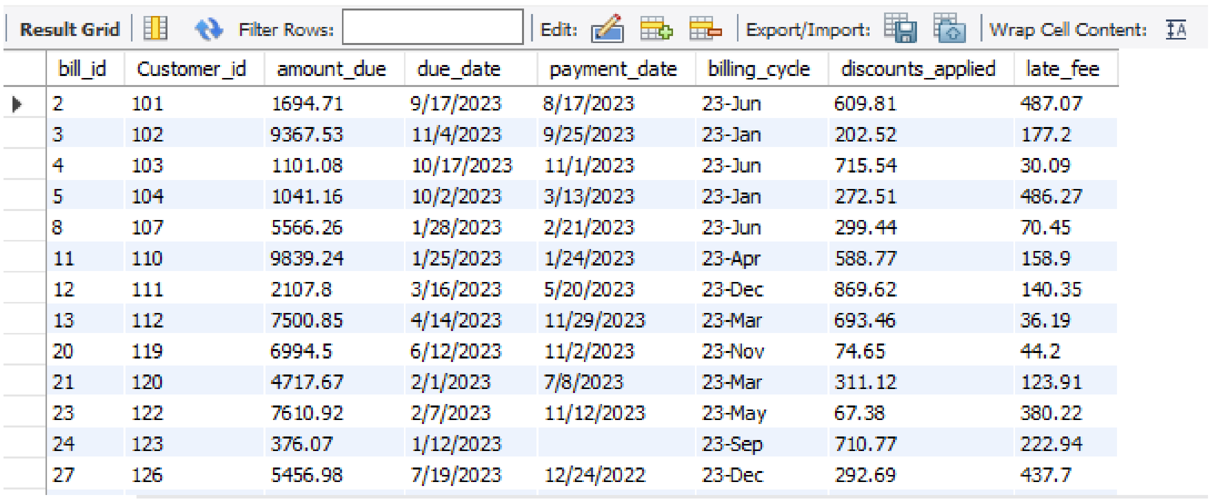
WHERE amount_due > 1000;



bill_id	Customer_id	amount_due	due_date	payment_date	billing_cycle	discounts_applied	late_fee
1	100	8262.08	5/4/2023	12/22/2023	23-Nov	103.71	955.9
2	101	1694.71	9/17/2023	8/17/2023	23-Jun	609.81	487.07
3	102	9367.53	11/4/2023	9/25/2023	23-Jan	202.52	177.2
4	103	1101.08	10/17/2023	11/1/2023	23-Jun	715.54	30.09
5	104	1041.16	10/2/2023	3/13/2023	23-Jan	272.51	486.27
6	105	6753.59	6/21/2023	1/13/2023	23-Apr	891.88	846.89
7	106	2737.5	6/26/2023		23-Jun	15.51	605.7
8	107	5566.26	1/28/2023	2/21/2023	23-Jun	299.44	70.45
9	108	3644.35	6/5/2023	11/1/2023	23-Dec	307.98	682.41
10	109	9717.41	9/4/2023		23-Feb	359.25	843.36
11	110	9839.24	1/25/2023	1/24/2023	23-Apr	588.77	158.9
12	111	2107.8	3/16/2023	5/20/2023	23-Dec	869.62	140.35
13	112	7500.85	4/14/2023	11/29/2023	23-Mar	693.46	36.19

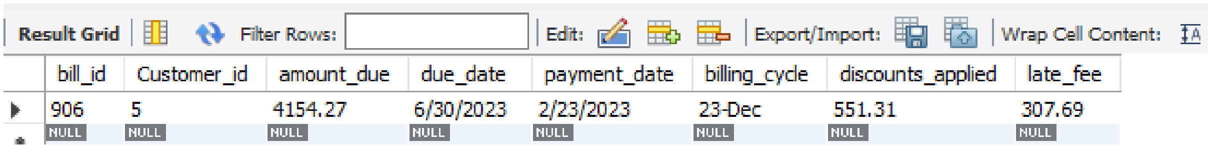
3. Exercise: Find all the late fee less than 500

*SELECT **
FROM billing
WHERE late_fee <500;


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

bill_id	Customer_id	amount_due	due_date	payment_date	billing_cycle	discounts_applied	late_fee
2	101	1694.71	9/17/2023	8/17/2023	23-Jun	609.81	487.07
3	102	9367.53	11/4/2023	9/25/2023	23-Jan	202.52	177.2
4	103	1101.08	10/17/2023	11/1/2023	23-Jun	715.54	30.09
5	104	1041.16	10/2/2023	3/13/2023	23-Jan	272.51	486.27
8	107	5566.26	1/28/2023	2/21/2023	23-Jun	299.44	70.45
11	110	9839.24	1/25/2023	1/24/2023	23-Apr	588.77	158.9
12	111	2107.8	3/16/2023	5/20/2023	23-Dec	869.62	140.35
13	112	7500.85	4/14/2023	11/29/2023	23-Mar	693.46	36.19
20	119	6994.5	6/12/2023	11/2/2023	23-Nov	74.65	44.2
21	120	4717.67	2/1/2023	7/8/2023	23-Mar	311.12	123.91
23	122	7610.92	2/7/2023	11/12/2023	23-May	67.38	380.22
24	123	376.07	1/12/2023		23-Sep	710.77	222.94
27	126	5456.98	7/19/2023	12/24/2022	23-Dec	292.69	437.7

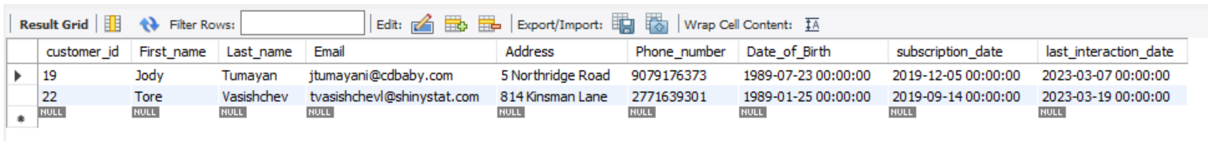
4. Exercise: Show bills that were generated for `customer_id' 5 .
Query:
*SELECT **
FROM billing
WHERE customer_id = '5';


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

bill_id	Customer_id	amount_due	due_date	payment_date	billing_cycle	discounts_applied	late_fee
906	5	4154.27	6/30/2023	2/23/2023	23-Dec	551.31	307.69
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Using WHERE with (IN, OR, AND, NOT EQUAL TO, NOT IN)

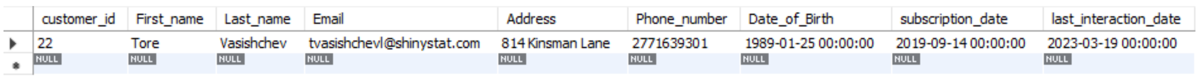
1. Exercise: Identify customer who live at either '5 Northridge Road', '814 Kinsman Lane'
Query:
*SELECT **
FROM customer
WHERE address IN ('5 Northridge Road', '814 Kinsman Lane');


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	First_name	Last_name	Email	Address	Phone_number	Date_of_Birth	subscription_date	last_interaction_date
19	Jody	Tumayan	jtumayani@cdbaby.com	5 Northridge Road	9079176373	1989-07-23 00:00:00	2019-12-05 00:00:00	2023-03-07 00:00:00
22	Tore	Vasishchev	tvasishchevl@shinystat.com	814 Kinsman Lane	2771639301	1989-01-25 00:00:00	2019-09-14 00:00:00	2023-03-19 00:00:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. **Exercise:** using or and AND

*SELECT **
FROM customer
WHERE address IN ('5 Northridge Road', '814 Kinsman Lane')   AND phone_number LIKE '277%';

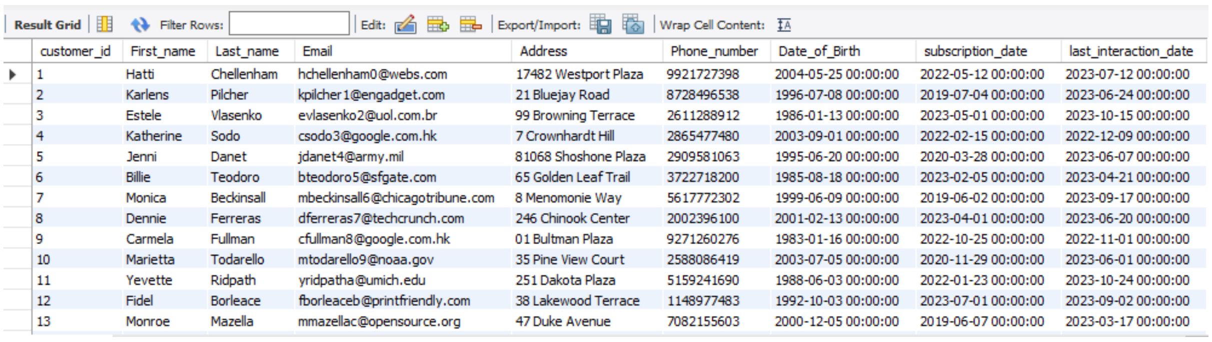


	customer_id	First_name	Last_name	Email	Address	Phone_number	Date_of_Birth	subscription_date	last_interaction_date
▶	22	Tore	Vasishchev	tvasishchevl@shinystat.com	814 Kinsman Lane	2771639301	1989-01-25 00:00:00	2019-09-14 00:00:00	2023-03-19 00:00:00
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Exercise: Display customer whose phone number is NOT '123-456-7890'.

Query:

*SELECT **
FROM customer
WHERE phone_number <> '123-456-7890';



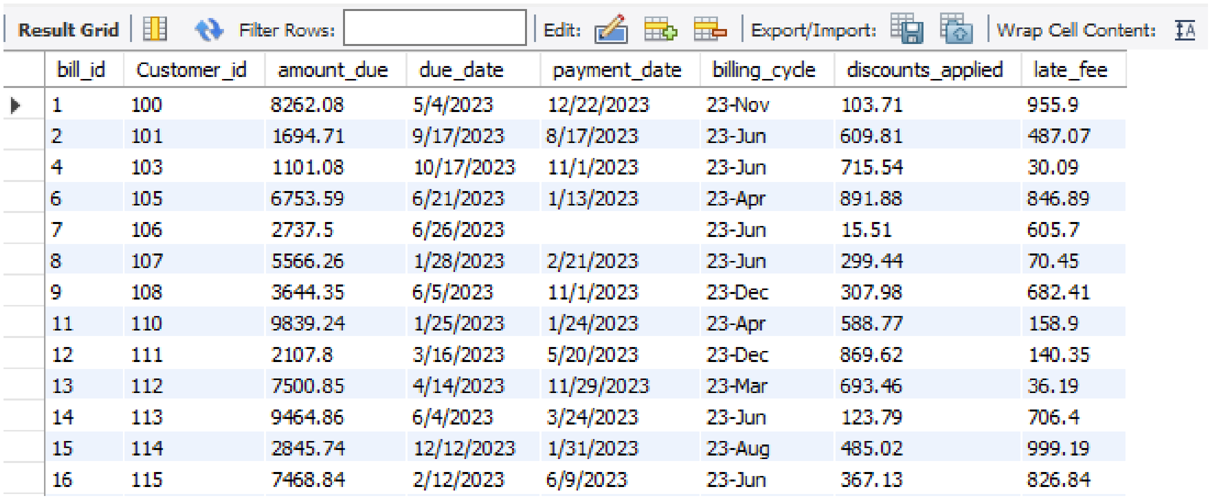
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	customer_id	First_name	Last_name	Email	Address	Phone_number	Date_of_Birth	subscription_date	last_interaction_date
▶	1	Hatti	Chellenham	hchellenham0@webs.com	17482 Westport Plaza	9921727398	2004-05-25 00:00:00	2022-05-12 00:00:00	2023-07-12 00:00:00
	2	Karlens	Pilcher	kpilcher1@engadget.com	21 Bluejay Road	8728496538	1996-07-08 00:00:00	2019-07-04 00:00:00	2023-06-24 00:00:00
	3	Estele	Vlasenko	evlasenko2@uol.com.br	99 Browning Terrace	2611288912	1986-01-13 00:00:00	2023-05-01 00:00:00	2023-10-15 00:00:00
	4	Katherine	Sodo	csodo3@google.com.hk	7 Crownhardt Hill	2865477480	2003-09-01 00:00:00	2022-02-15 00:00:00	2022-12-09 00:00:00
	5	Jenni	Danet	jdanet4@army.mil	81068 Shoshone Plaza	2909581063	1995-06-20 00:00:00	2020-03-28 00:00:00	2023-06-07 00:00:00
	6	Billie	Teodoro	bteodoro5@sfgate.com	65 Golden Leaf Trail	3722718200	1985-08-18 00:00:00	2023-02-05 00:00:00	2023-04-21 00:00:00
	7	Monica	Beckinsall	mbeckinsall6@chicagotribune.com	8 Menomonie Way	5617772302	1999-06-09 00:00:00	2019-06-02 00:00:00	2023-09-17 00:00:00
	8	Dennie	Ferreras	dferreras7@techcrunch.com	246 Chinook Center	2002396100	2001-02-13 00:00:00	2023-04-01 00:00:00	2023-06-20 00:00:00
	9	Carmela	Fullman	cfullman8@google.com.hk	01 Bultman Plaza	9271260276	1983-01-16 00:00:00	2022-10-25 00:00:00	2022-11-01 00:00:00
	10	Marietta	Todarello	mtodarello9@noaa.gov	35 Pine View Court	2588086419	2003-07-05 00:00:00	2020-11-29 00:00:00	2023-06-01 00:00:00
	11	Yevette	Ridpath	yridpatha@umich.edu	251 Dakota Plaza	5159241690	1988-06-03 00:00:00	2022-01-23 00:00:00	2023-10-24 00:00:00
	12	Fidel	Borleace	fborleaceb@printfriendly.com	38 Lakewood Terrace	1148977483	1992-10-03 00:00:00	2023-07-01 00:00:00	2023-09-02 00:00:00
	13	Monroe	Mazella	mmazellac@opensource.org	47 Duke Avenue	7082155603	2000-12-05 00:00:00	2019-06-07 00:00:00	2023-03-17 00:00:00

4. Exercise: List all bills except those with billing cycles in "January 2023" and "February 2023".

Query:

*SELECT **
FROM billing
WHERE billing_cycle NOT IN ('23-Jan', '23-Feb');



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

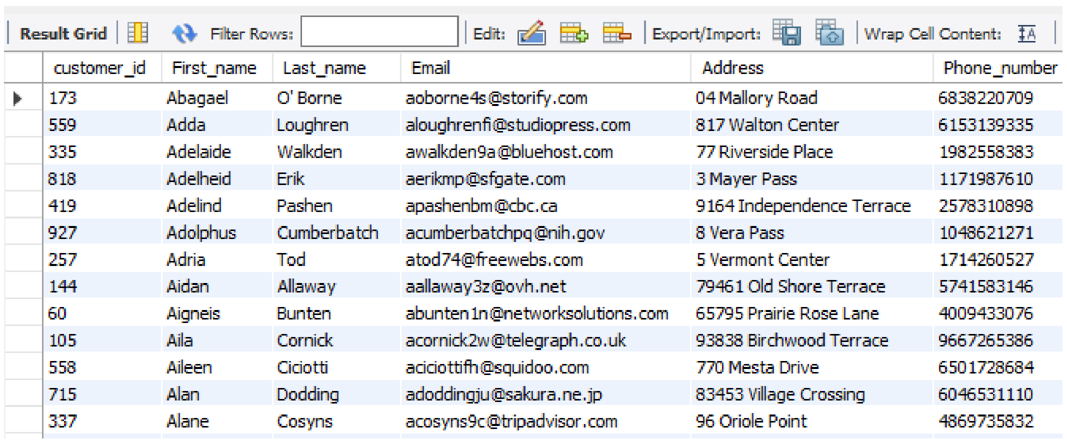
	bill_id	Customer_id	amount_due	due_date	payment_date	billing_cycle	discounts_applied	late_fee
▶	1	100	8262.08	5/4/2023	12/22/2023	23-Nov	103.71	955.9
	2	101	1694.71	9/17/2023	8/17/2023	23-Jun	609.81	487.07
	4	103	1101.08	10/17/2023	11/1/2023	23-Jun	715.54	30.09
	6	105	6753.59	6/21/2023	1/13/2023	23-Apr	891.88	846.89
	7	106	2737.5	6/26/2023		23-Jun	15.51	605.7
	8	107	5566.26	1/28/2023	2/21/2023	23-Jun	299.44	70.45
	9	108	3644.35	6/5/2023	11/1/2023	23-Dec	307.98	682.41
	11	110	9839.24	1/25/2023	1/24/2023	23-Apr	588.77	158.9
	12	111	2107.8	3/16/2023	5/20/2023	23-Dec	869.62	140.35
	13	112	7500.85	4/14/2023	11/29/2023	23-Mar	693.46	36.19
	14	113	9464.86	6/4/2023	3/24/2023	23-Jun	123.79	706.4
	15	114	2845.74	12/12/2023	1/31/2023	23-Aug	485.02	999.19
	16	115	7468.84	2/12/2023	6/9/2023	23-Jun	367.13	826.84

Using ORDER BY

1. Exercise: Order customer by their names in ascending order.

Query:

*SELECT **
FROM customer
ORDER BY first_name ASC;



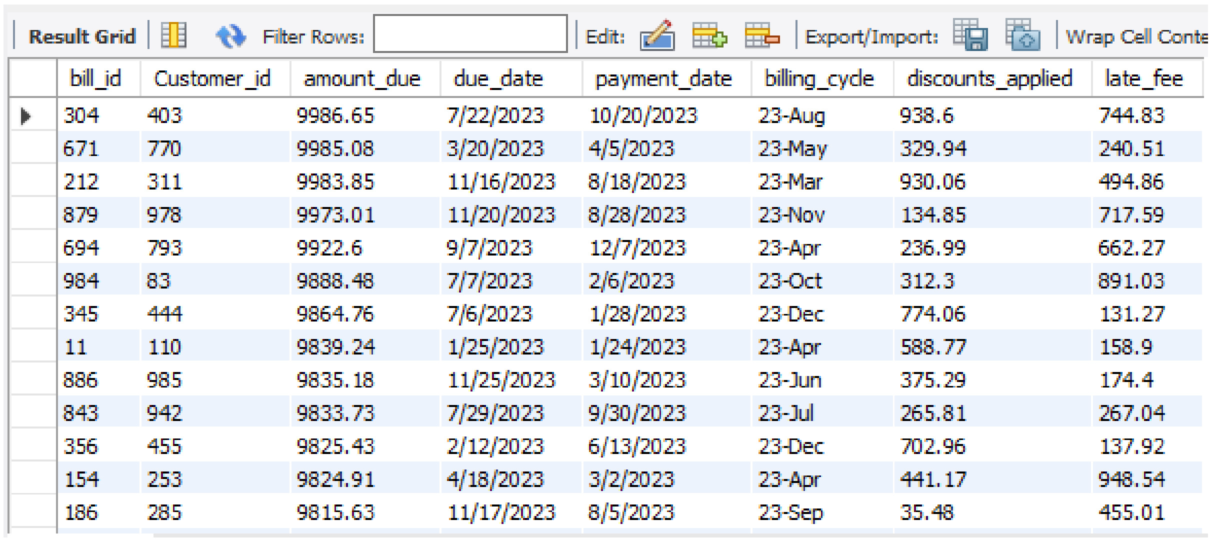
customer_id	First_name	Last_name	Email	Address	Phone_number
173	Abagael	O' Borne	aoborne4s@storify.com	04 Mallory Road	6838220709
559	Adda	Loughren	aloughrenfi@studiopress.com	817 Walton Center	6153139335
335	Adelaide	Walkden	awalkden9a@bluehost.com	77 Riverside Place	1982558383
818	Adelheid	Erik	aerikmp@sfgate.com	3 Mayer Pass	1171987610
419	Adelind	Pashen	apashenbm@cbc.ca	9164 Independence Terrace	2578310898
927	Adolphus	Cumberbatch	acumberbatchpq@nih.gov	8 Vera Pass	1048621271
257	Adria	Tod	atod74@freewebs.com	5 Vermont Center	1714260527
144	Aidan	Allaway	aallaway3z@ovh.net	79461 Old Shore Terrace	5741583146
60	Aigneis	Bunten	abunten1n@networksolutions.com	65795 Prairie Rose Lane	4009433076
105	Aila	Cornick	acornick2w@telegraph.co.uk	93838 Birchwood Terrace	9667265386
558	Aileen	Ciciotti	aciciottifh@squidoo.com	770 Mesta Drive	6501728684
715	Alan	Dodding	adoddingju@sakura.ne.jp	83453 Village Crossing	6046531110
337	Alane	Cosyns	acosyns9c@tripadvisor.com	96 Oriole Point	4869735832

2. Exercise: Display bills from the `billing` table ordered by `amount_due` in descending order.

Query:

*SELECT **

FROM billing ORDER BY amount_due DESC;



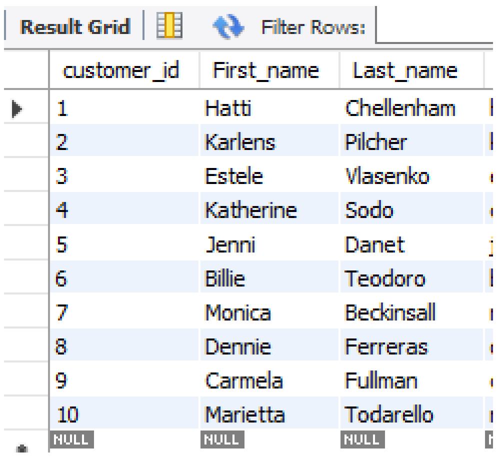
bill_id	Customer_id	amount_due	due_date	payment_date	billing_cycle	discounts_applied	late_fee
304	403	9986.65	7/22/2023	10/20/2023	23-Aug	938.6	744.83
671	770	9985.08	3/20/2023	4/5/2023	23-May	329.94	240.51
212	311	9983.85	11/16/2023	8/18/2023	23-Mar	930.06	494.86
879	978	9973.01	11/20/2023	8/28/2023	23-Nov	134.85	717.59
694	793	9922.6	9/7/2023	12/7/2023	23-Apr	236.99	662.27
984	83	9888.48	7/7/2023	2/6/2023	23-Oct	312.3	891.03
345	444	9864.76	7/6/2023	1/28/2023	23-Dec	774.06	131.27
11	110	9839.24	1/25/2023	1/24/2023	23-Apr	588.77	158.9
886	985	9835.18	11/25/2023	3/10/2023	23-Jun	375.29	174.4
843	942	9833.73	7/29/2023	9/30/2023	23-Jul	265.81	267.04
356	455	9825.43	2/12/2023	6/13/2023	23-Dec	702.96	137.92
154	253	9824.91	4/18/2023	3/2/2023	23-Apr	441.17	948.54
186	285	9815.63	11/17/2023	8/5/2023	23-Sep	35.48	455.01

Using LIMIT

1. Exercise: Show only the first 10 customer.

Query:

*SELECT * FROM customer LIMIT 10;*



	customer_id	First_name	Last_name
▶	1	Hatti	Chellenham
	2	Karlens	Pilcher
	3	Estele	Vlasenko
	4	Katherine	Sodo
	5	Jenni	Danet
	6	Billie	Teodoro
	7	Monica	Beckinsall
	8	Dennie	Ferreras
	9	Carmela	Fullman
	10	Marietta	Todarello
*	NULL	NULL	NULL

2. Exercise: List the top 5 highest bills from the `billing` table.

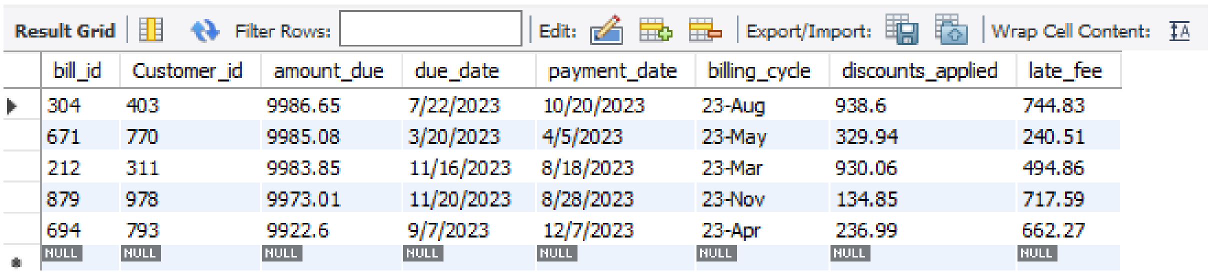
Query:

*SELECT **

FROM billing

ORDER BY amount_due DESC

LIMIT 5;



	bill_id	Customer_id	amount_due	due_date	payment_date	billing_cycle	discounts_applied	late_fee
▶	304	403	9986.65	7/22/2023	10/20/2023	23-Aug	938.6	744.83
	671	770	9985.08	3/20/2023	4/5/2023	23-May	329.94	240.51
	212	311	9983.85	11/16/2023	8/18/2023	23-Mar	930.06	494.86
	879	978	9973.01	11/20/2023	8/28/2023	23-Nov	134.85	717.59
	694	793	9922.6	9/7/2023	12/7/2023	23-Apr	236.99	662.27
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Exercise: Retrieve the latest 3 bills based on the due date.

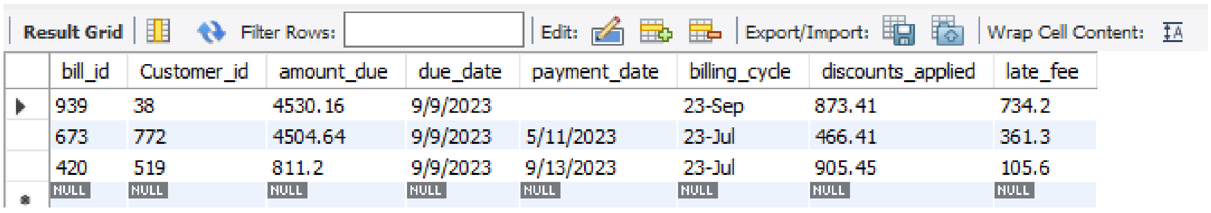
Query:

*SELECT **

FROM billing

ORDER BY due_date DESC

LIMIT 3;



	bill_id	Customer_id	amount_due	due_date	payment_date	billing_cycle	discounts_applied	late_fee
▶	939	38	4530.16	9/9/2023		23-Sep	873.41	734.2
	673	772	4504.64	9/9/2023	5/11/2023	23-Jul	466.41	361.3
	420	519	811.2	9/9/2023	9/13/2023	23-Jul	905.45	105.6
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL