

## **CREATE SCHEMA 'project\_emails'**

CREATING DATABASE TABLES:

CREATE TABLE account

```
(  
Account_ID int NOT NULL AUTO_INCREMENT PRIMARY KEY,  
Email varchar(250),  
Password varchar(40),  
First_Name varchar(40),  
Last_Name varchar(40),  
Email_Open DATE,  
Email_Close DATE,  
Password_Expire DATE,  
UNIQUE (Email)  
);
```

CREATE TABLE Email

```
(  
Email_ID int NOT NULL AUTO_INCREMENT PRIMARY KEY,  
subject VARCHAR(255),  
sender INTEGER NOT NULL  
FOREIGN KEY (sender) REFERENCES Account(Account_ID)  
);
```

CREATE TABLE Email\_Recipients (

```
email_id INTEGER NOT NULL,  
recipient VARCHAR(255) NOT NULL,  
FOREIGN KEY (email_id) REFERENCES Email(Email_ID)  
);
```

CREATE TABLE Email\_content

```
(  
email_id int NOT NULL ,  
email_message varchar(1000),  
attachment int,  
size int,  
sent_Date DATE,  
foreign key(Email_ID) REFERENCES Email(Email_ID)  
);
```

CREATE TABLE Mailbox (

```
Mail_id INTEGER PRIMARY KEY AUTOINCREMENT,  
account_id INTEGER NOT NULL,  
FOREIGN KEY (account_id) REFERENCES Account(id)  
);
```

```
CREATE TABLE Inbox  
(  
  inbox_id INTEGER NOT NULL,  
  email_id INTEGER NOT NULL,  
  mailbox_id INTEGER NOT NULL,  
  FOREIGN KEY (email_id) REFERENCES Email(id),  
  FOREIGN KEY (mailbox_id) REFERENCES Mailbox(id)  
);
```

```
CREATE TABLE Sent  
(  
  sent_id INTEGER NOT NULL,  
  email_id INTEGER NOT NULL,  
  mailbox_id INTEGER NOT NULL,  
  FOREIGN KEY (email_id) REFERENCES Email(id),  
  FOREIGN KEY (mailbox_id) REFERENCES Mailbox(id)  
);
```

```
CREATE TABLE Junk  
(  
  junk_id INTEGER NOT NULL,  
  email_id INTEGER NOT NULL,  
  mailbox_id INTEGER NOT NULL,  
  FOREIGN KEY (email_id) REFERENCES Email(id),  
  FOREIGN KEY (mailbox_id) REFERENCES Mailbox(id)  
);
```

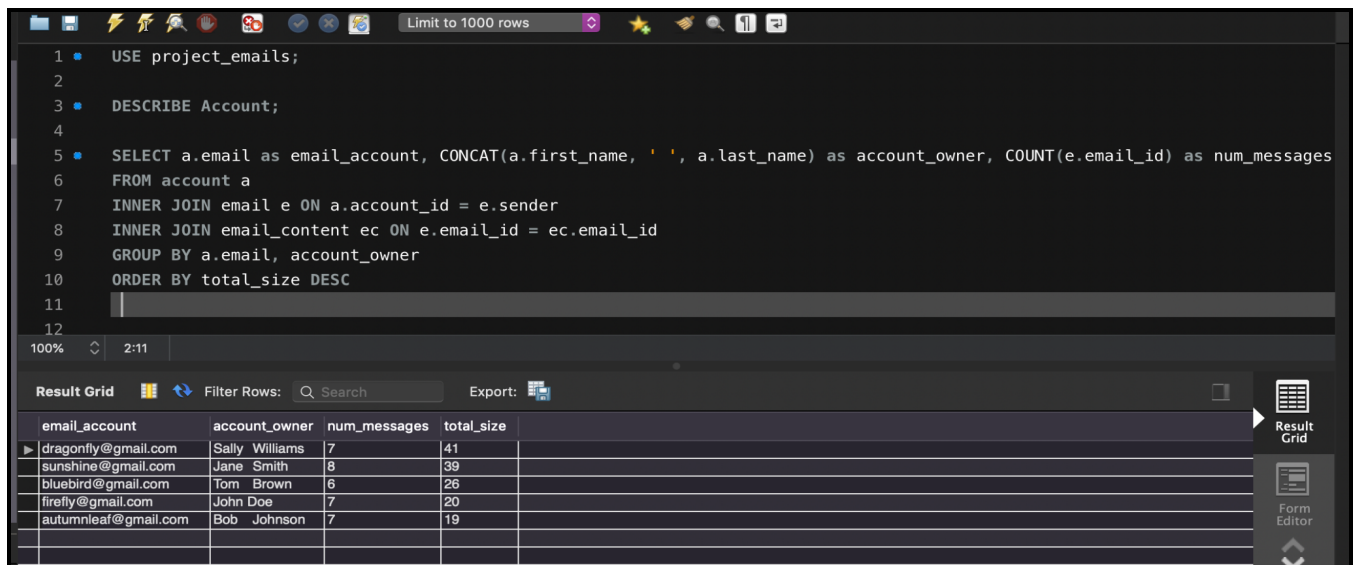
### **Outline for the Database:**

- The Account table stores information about email accounts, including the account owner's email address, password, and name. It also has fields for tracking when the account was opened and closed, as well as when the password is set to expire.
- The Email table stores information about individual emails, including the subject and the sender's account ID.

- The Email\_Recipients table stores a list of recipients for each email, with the email ID as a foreign key to the Email table.
- The Email\_content table stores the contents of each email, including the message and any attachments. It also has fields for storing the size of the email and the date it was sent.
- The Mailbox table stores information about email accounts, including the account ID as a foreign key to the Account table.
- The Inbox, Sent, and Junk tables store information about emails that have been organized into these different categories, including the email ID and the mailbox ID as foreign keys to the Email and Mailbox tables, respectively. BUT IN THE FUTURE I WISH I ONLY CREATED ONE TABLE FOLDER WITH A TYPE so then it easier to find with sql commands!

## QUESTIONS:

1. Identify the number of messages and total size by email accounts. Display four columns: email account, account owner name, number of messages, and total size. Display the account with the greatest storage requirements first



The screenshot shows a SQL IDE interface. The top panel contains a SQL query. The bottom panel displays the results in a table format. The query is as follows:

```
1 • USE project_emails;
2
3 • DESCRIBE Account;
4
5 • SELECT a.email as email_account, CONCAT(a.first_name, ' ', a.last_name) as account_owner, COUNT(e.email_id) as num_messages
6 FROM account a
7 INNER JOIN email e ON a.account_id = e.sender
8 INNER JOIN email_content ec ON e.email_id = ec.email_id
9 GROUP BY a.email, account_owner
10 ORDER BY total_size DESC
11
12
```

The results table has the following columns: email\_account, account\_owner, num\_messages, and total\_size. The data is sorted by total\_size in descending order.

email_account	account_owner	num_messages	total_size
dragonfly@gmail.com	Sally Williams	7	41
sunshine@gmail.com	Jane Smith	8	39
bluebird@gmail.com	Tom Brown	6	26
firefly@gmail.com	John Doe	7	20
autumnleaf@gmail.com	Bob Johnson	7	19

2. Send a new message to "firefly@gmail.com". Identify the SQL operations to create and send the message. In addition, Display the new message in the SENT folder and the new message in John Doe's INBOX folder.

[Note: So first I need to manually add the email and create it using SQL operations and this is specified for my database]

```
INSERT INTO email (subject, sender) VALUES ('Question 2!', (SELECT account_id FROM account WHERE email = 'firefly@gmail.com'));
```

```
INSERT INTO email_recipients (email_id, recipient) VALUES ((SELECT email_id FROM email WHERE subject = 'Question 2!'), 'firefly@gmail.com');
```

```
INSERT INTO email_content (email_id, email_message, attachment, size, sent_date, email_content_id) VALUES ((SELECT email_id FROM email WHERE subject = 'Question 2!'), 'This is correct!', 1, 100, '1/15/2022', 100);
```

OUTPUT SHOWS THESE COMMANDS WENT THROUGH:

707	13:12:44	INSERT INTO email (subject, sender) VALUES ('Question 2!', (SELECT account_id FROM account WHERE email = 'firefly@gmail.com'));	1 row(s) affected	0.0033 sec
708	13:12:59	INSERT INTO email_content (email_id, email_message, attachment, size, sent_date) VALUES ((SELECT email_id FROM email WHERE subject = 'Question 2!'), 'This is correct!', 1, 100, '1/15/2022');	Error Code: 1364. Field 'email_content_id' doesn't have a default value	0.0012 sec
709	13:13:45	SELECT * FROM project_emails.Email_Content LIMIT 0, 1000	35 row(s) returned	0.00030 sec / 0.000...
710	13:14:09	INSERT INTO email_content (email_id, email_message, attachment, size, sent_date) VALUES ((SELECT email_id FROM email WHERE subject = 'Question 2!'), 'This is correct!', 1, 100, '1/15/2022');	Error Code: 1364. Field 'email_content_id' doesn't have a default value	0.00051 sec
711	13:15:02	INSERT INTO email_content (email_id, email_message, attachment, size, sent_date) VALUES ((SELECT email_id FROM email WHERE subject = 'Question 2!'), 'This is correct!', 1, 100, '1/15/2022');	Error Code: 1136. Column count doesn't match value count	0.00036 sec
712	13:15:25	INSERT INTO email_content (email_id, email_message, attachment, size, sent_date) VALUES ((SELECT email_id FROM email WHERE subject = 'Question 2!'), 'This is correct!', 1, 100, '1/15/2022');	Error Code: 1364. Field 'email_content_id' doesn't have a default value	0.00051 sec
713	13:16:04	INSERT INTO email_content (email_id, email_message, attachment, size, sent_date, email_content_id) VALUES ((SELECT email_id FROM email WHERE subject = 'Question 2!'), 'This is correct!', 1, 100, '1/15/2022', 100);	1 row(s) affected	0.0040 sec

[Then the next step that I did was handling the location of this new email that was inserted into the database. The email that I sent was connected to John Doe so that's why I used it for the mailbox]

```
SELECT e.subject, ec.email_message, ec.attachment, ec.size, ec.sent_date
FROM inbox i
INNER JOIN email e ON i.email_id = e.email_id
INNER JOIN email_content ec ON e.email_id = ec.email_id
INNER JOIN account a ON i.account_id = a.account_id
WHERE a.first_name = 'John' AND a.last_name = 'Doe'
```

3. Move a message from the INBOX to the FRIENDS folder for the account firefly@gmail.com. Identify the SQL operations required to implement. In addition, display the INBOX and SENT folders before and after the operation.

[Note: So, the first thing that needs to be done is to create a new table called friends that is connected to the mailbox just like inbox, sent, and junk]

```
CREATE TABLE Friends (
  account_id INTEGER NOT NULL,
  email_id INTEGER NOT NULL,
  mailbox_id INTEGER NOT NULL,
  FOREIGN KEY (account_id) REFERENCES Account(id),
  FOREIGN KEY (email_id) REFERENCES Email(id),
  FOREIGN KEY (mailbox_id) REFERENCES Mailbox(id)
);
```

inbox_id	mailbox_id	email_id	
1	1	1	
2	2	2	
3	3	3	
4	4	4	
5	5	5	
NULL	NULL	NULL	

[Note we are going to use the email\_id "1" to move out of the inbox folder and into the friend's table]

```
INSERT INTO Friends (account_id, email_id, mailbox_id)
SELECT account_id, email_id, mailbox_id
FROM Inbox
WHERE email_id = 1;
```

Friends_id	mailbox_id	email_id	
1	1	1	
NULL	NULL	NULL	

4. Create a SQL view to display all messages in the INBOX folder for "firefly@gmail.com". Display the columns Received Date, FROM, TO, and Subject. Display the most recent message first.

[Note for my database I handled “to” using email\_recipients, I used from as “sender”]

```
CREATE VIEW InboxMessages AS
SELECT Email_content.sent_Date AS Received_Date, Email.sender AS sender, Email_Recipients.recipient AS recipient, Email.subject AS Subject
FROM Email
INNER JOIN Email_content ON Email.Email_ID = Email_content.Email_ID
INNER JOIN Email_Recipients ON Email.Email_ID = Email_Recipients.Email_ID
INNER JOIN Inbox ON Email.Email_ID = Inbox.email_id
INNER JOIN Mailbox ON Inbox.mailbox_id = Mailbox.Mail_id
INNER JOIN Account ON Mailbox.account_id = Account.Account_ID
WHERE Account.Email = 'firefly@gmail.com'
ORDER BY Email_content.sent_Date DESC
```

[Note: InboxMessages view only displays messages that are in the INBOX folder for "firefly@gmail.com". Not including messages from other email accounts or messages in other folders, IT RUNS!!! COMMANDS WORK but there aren't any emails from firefly in my databox in the inbox so the table is empty]

```
4
5 • SELECT * FROM InboxMessages;
6 • SELECT * FROM InboxMessages ORDER BY Received_Date ASC;
7 • SELECT * FROM InboxMessages ORDER BY Subject DESC;
8
9
```

1:10

Result Grid Filter Rows: Search Export:

Received_Date	sender	recipient	Subject
---------------	--------	-----------	---------

5. Identify all messages with the term love in the subject or body. Display the columns Received Date, FROM, TO, and Subject. Display the most recent message first.

[Note: “from” is the sender and “to” is the recipient. Finally, please be aware that my database only has one email with any love in the subject or contents]

```
3 • SELECT Email_content.sent_Date AS Received_Date, Email.sender AS sender, Email_Recipients.recipient AS recipient, Email.subject
4 FROM Email
5 INNER JOIN Email_content ON Email.Email_ID = Email_content.Email_ID
6 INNER JOIN Email_Recipients ON Email.Email_ID = Email_Recipients.Email_ID
7 WHERE Email.subject LIKE '%love%' OR Email_content.email_message LIKE '%love%'
8 ORDER BY Email_content.sent_Date DESC
9
```

38:8

Result Grid Filter Rows: Search Export:


Received_Date	sender	recipient	Subject
1/2/2022	2	firefly@gmail.com	love

6. Delete one message. Discuss your approach to deleting messages and identify the SQL operations required to implement. Display the folder before and after the deletion

[Note Deleting one email address means it would need to be completely wiped out of the database so it means going through all the tables including email, and email contents. Lastly, I choose an email where we can see this change occur with the email located in the inbox with email id=3. As we can see the before and after of the inbox it was deleted from the folder]

```
DELETE FROM Email WHERE Email_ID = 3;  
DELETE FROM Email_content WHERE Email_ID = 3;  
DELETE FROM Inbox WHERE email_id = 3;
```



	inbox_id	mailbox_id	email_id
▶	1	1	1
	2	2	2
	3	3	3
	4	4	4
	5	5	5

7  **SELECT \* FROM INBOX**

8

9

20:7

**Result Grid**   Filter Rows:

inbox_id	mailbox_id	email_id
1	1	1
2	2	2
4	4	4
5	5	5
NULL	NULL	NULL

7. Identify and move spam messages to the JUNK folder. Identify the SQL operations required to implement.

[Note so first I as a programmer must find out what can be considered a spam message, so for this, I input some emails that I considered spam to test this question out. I will look at the number of emails that have "win" in them. But it will leave the "junk\_id" column as NULL. Since the "junk\_id" column is an AUTO\_INCREMENT column, it will automatically generate a unique value for each row inserted into the table]

```
INSERT INTO Junk (junk_id, email_id, mailbox_id)
SELECT NULL, Email.Email_ID, Mailbox.Mail_id
FROM Email
JOIN Mailbox ON Email.sender = Mailbox.account_id
WHERE Email.subject LIKE '%win%';
```

junk_id	mailbox_id	email_id	
1	3	5	
2	3	10	
3	3	29	
4	5	36	
NULL	NULL	NULL	

8. Identify accounts that have not received new messages today. Display the email address.



[Note: I put today as "12/14/2022", the SQL Commands as well as the Output below]

```
1  USE project_emails;
2
3  •  SELECT Account.Email
4     FROM Account
5     LEFT JOIN Email_content ON Account.Account_ID = Email_content.Email_ID
6     WHERE Email_content.sent_Date != "12/14/2022";
7
```

100% 1:7

Result Grid Filter Rows: Search Export:

Email
▶ dragonfly@gmail.com
firefly@gmail.com
sunshine@gmail.com

9. Utilize security roles to limit users to edit/view rights to their mailbox, but don't have access to other mailboxes. Identify the SQL operations to implement and demonstrate the functionality of the security roles

```
CREATE ROLE user1_mailbox;
GRANT SELECT, INSERT, UPDATE, DELETE ON Mailbox TO user1_mailbox;
```

[Note: To allow users to edit their own mailbox as well, you can grant the SELECT, INSERT, UPDATE, and DELETE privileges to the security role ]

10. Utilize security roles to allow the administrator to view all mailboxes, but can't change any messages. Identify the SQL operations to implement and demonstrate the functionality of the security roles.

```
use project_emails;

CREATE ROLE admin_mailbox;
GRANT SELECT ON Mailbox TO admin_mailbox;
CREATE VIEW admin_email_view AS
SELECT *
FROM Email;

GRANT SELECT ON admin_email_view TO admin_mailbox;
REVOKE INSERT, UPDATE, DELETE ON admin_email_view FROM admin_mailbox;
```

[Note: includes all rows from the "Email" table, but only allows the SELECT privilege on the view. This will allow the administrator to view all emails, but not change or delete them.]

11. Identify the SQL operations to implement and demonstrate the functionality of the security roles

[Note: They use security roles and privileges to control what users can do. Security roles are important in an email database because they allow you to control what users can do with their accounts and protect the data in the database. Without security roles, users may be able to access and modify data that they are not supposed to, which can compromise the security and integrity of the database. Use security roles to limit users to only view or edit their own mailbox, and prevent them from accessing or modifying the mailboxes of other users. This helps to protect the privacy of each user's data and ensures that only the user has control over their own mailbox. An important tool for protecting the data in an email database and ensuring that it is only accessed and modified by authorized users]

- CREATE ROLE - Use this operation to create a new security role
- GRANT - Use this operation to grant privileges for tables or views
- REVOKE - Use this operation to revoke privileges
- GRANT ROLE - assign a security role to a user
- REVOKE ROLE - remove a security role from a user

12. In one SQL window, move message 1 to a new folder. Don't commit. In another SQL window, move message 1 to a different folder. Don't commit. Explain your results. Resolve the problem. Disable the auto-commit flag at the top of the

window before performing this operation.

[Note: Part one the "email\_id" of 4 to "Junk" and then "email\_id" of 5 to "Friends". Once both of these two inquiries are run in separate SQL windows and I did not commit the work. It still doesn't update when I do the second one, basically it's like I never had input the first command in the first place. the changes are visible only to me, this happened a lot with the project where I had to input data many times because I had forgot to commit and it would just get erased in the process. ]

```
1  use project_emails;
2
3  INSERT INTO Junk (junk_id, email_id, mailbox_id)
4  SELECT NULL, Email.Email_ID, Mailbox.Mail_id
5  FROM Email
6  JOIN Mailbox ON Email.sender = Mailbox.account_id
7  WHERE Email.Email_ID = 4;
```

```
use project_emails;

INSERT INTO Friends (friend_id, email_id, mailbox_id)
SELECT NULL, Email.Email_ID, Mailbox.Mail_id
FROM Email
JOIN Mailbox ON Email.sender = Mailbox.account_id
WHERE Email.Email_ID = 5;
```

13. In one SQL window, delete all messages for one account. Don't commit. In another SQL window, move all messages for the same account to the FRIENDS folder. Don't commit. Explain your results. Resolve the problem. Create a backup of your table before implementing. To create a backup table, enter CREATE TABLE AS SELECT \* FROM; COMMIT; Then you can

rename a table using the RENAME TABLE commit. Disable the auto-commit flag at the top of the window before performing this operation.

[Note: I am using account\_id=2 as the same account to delete all their messages and then move them into the friends table/folder under a different SQL window. But I didn't commit anything, my results did run without error. But there were no emails in the friends table/folder because I had deleted them earlier. To resolve this issue a backup table will be created called "recovery\_emails" which will be used like in normal programming as a temp value but in this case we will use it when transferring emails that will be later on deleted from the database so it can be a smooth recovery without any data loss ]

```
1 • use project_emails;
2
3 • DELETE Email, Junk, Sent, Inbox
4 FROM Email
5 LEFT JOIN Junk ON Email.Email_ID = Junk.email_id
6 LEFT JOIN Sent ON Email.Email_ID = Sent.email_id
7 LEFT JOIN Inbox ON Email.Email_ID = Inbox.email_id
8 WHERE Email.sender = 2;
9
```

```
use project_emails;

INSERT INTO Friends (friend_id, email_id, mailbox_id)
SELECT NULL, Email.Email_ID, Mailbox.Mail_id
FROM Email
JOIN Mailbox ON Email.sender = Mailbox.account_id
WHERE Email.sender = 2;
```

```
CREATE TABLE recovery_emails
(
  Email_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  Subject VARCHAR(255),
  Sender INT NOT NULL,
  FOREIGN KEY (Sender) REFERENCES Account(Account_ID)
);
```

14. In one SQL window, change the password for account 1. Don't commit. In another SQL window, change the last name of the owner of the account Don't commit. Quit both Oracle sessions. Login to Oracle and display all information. Explain your results. Disable the auto-commit flag at the top of the windows before performing this operation

[Note I changed the password for account=5 in one window then changed the account=5 owners last name to "new last name" I didn't commit either changes. I quit the Oracle sessions and nothing saved until what I last committed. Nothing has changed. Every table contents remained the same despite the changes I've made, if the auto-commit flag on top were turned on then this would of been automatically saved once I had quit the sessions]

```
use project_emails;

UPDATE Account
SET Password = 'NewPasswordQ14'
WHERE Account_ID = 5;
```

```
use project_emails;  
  
UPDATE Account  
SET Last_Name = 'NewLastName'  
WHERE Account_ID = 5;
```

15. Use the SQL DESCRIBE operation to list the table structure for all tables.

[Note: SOME CHANGES MAY NOT APPEAR BELOW BECAUSE SOME QUESTIONS I DID NOT COMMIT INTO THE TABLES THEREFORE IT DIDN'T SAVE]

```
SHOW TABLES;  
DESCRIBE Account;  
DESCRIBE Email;  
DESCRIBE Email_Contents;  
DESCRIBE Friends;  
DESCRIBE Inbox;  
DESCRIBE Junk;  
DESCRIBE MAILBOX;  
DESCRIBE SENT;
```

OUTPUTS:

Account_ID	Email	Password	First_Name	Last_Name	Email_Open	Email_Close	Password_Expire_Date	
1	firefly@gmail.com	sunshine	John	Doe	01/01/2009	03/3/2022	01/01/2022	
2	sunshine@gmail.com	firefly	Jane	Smith	02/14/2008	NULL	02/14/2022	
3	autumnleaf@gmail.com	dragonfly	Bob	Johnson	02/15/2009	NULL	03/31/2022	
4	dragonfly@gmail.com	autumnleaf	Sally	Williams	03/24/2009	08/3/2022	04/15/2022	
5	bluebird@gmail.com	bluebird	Tom	Brown	04/20/2009	NULL	05/29/2022	
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

Email_ID	subject	sender
1	love	2
2	care	1
4	romance	4
5	win a house	3
6	job	2
7	vacation pay	1
8	hello	5
9	miss you!	4
10	win a car	3
11	can't wait	2
12	jobs	1
13	event	4
14	funeral	3
15	family event	2
16	movie date	1
17	stop the abuse	5
18	protect our planet	4
19	filming on set	3
20	we don't care	2
21	sometimes it is possible	1
22	we love	5
23	we want you	4
24	job offer	3
25	cancel subscription	2

email_id	email_message	attachment	size	sent_date	email_content_id
5	hello!	1	13	12/14/2022	1
6	bro	3	30	12/14/2022	2
4	I became paranoid that the school of jellyfish	3	30	11/9/2022	3
29	The anaconda was the greatest criminal	2	1	7/02/2021	4
12	He felt that dining on the bridge	1	1	3/2/2021	5
11	When he had to picnic on the beach,	0	1	3/18/2020	6
10	Thigh-high in the water,	0	1	3/10/2010	7
8	We have a lot of rain in June.	0	1	2/23/2009	8
14	Patricia found the meaning of life in a bowl	0	1	3/23/2010	9
22	The fence was confused about whether	0	1	4/4/2014	10
25	Poison ivy grew through the fence	0	1	4/9/2015	11
19	Various sea birds are elegant, but nothing is as elegant	0	1	4/20/2016	12
26	He was willing to find the depths of the rabbit hole in	0	1	5/24/2015	13
31	100 years old is such a young age if you happen to be	0	1	8/20/2018	14
21	There aren't enough towels in the world	0	1	4/29/2017	15
23	At that moment I was the most fearsome weasel	2	1	4/9/2011	16
24	I was offended by the suggestion tha	2	1	4/9/2013	17
34	She was amazed by the large chunks of ice washing	1	1	8/30/2021	18
27	I'd rather be a bird than a fish.	1	1	5/29/2021	19
20	Flesh-colored yoga pants were far worse than	3	1	4/23/2021	20
35	The blinking lights of the antenna tower came	3	2	9/9/2022	21
9	He went back to the video to see what had	2	1	2/3/2022	22
1	Homesickness became contagious in the young	1	1	1/2/2022	23
15	After fighting off the alligator, Brian still had to face	1	2	3/3/2022	24



	email_id	recipient	email_rep_id	
▶	1	firefly@gmail.com	1	
	2	firefly@gmail.com	2	
	6	firefly@gmail.com	3	
	3	firefly@gmail.com	4	
	4	firefly@gmail.com	5	
	3	firefly@gmail.com	6	
	3	bluebird@gmail.com	7	
	3	dragonfly@gmail.com	8	
	NULL	NULL	NULL	

	Friends_id	mailbox_id	email_id	
▶	1	1	1	
	NULL	NULL	NULL	

	inbox_id	mailbox_id	email_id	
▶	1	1	1	
	2	2	2	
	4	4	4	
	5	5	5	
	NULL	NULL	NULL	



Result Grid			
Filter Rows:			
Search			
Edit			
junk_id	mailbox_id	email_id	
▶ 1	3	5	
2	3	10	
3	3	29	
4	5	36	
NULL	NULL	NULL	

mail_id	account_id	
▶ 5	1	
4	2	
3	3	
2	4	
1	5	
NULL	NULL	

	sent_id	mailbox_id	email_id	
▶	1	1	30	
	2	2	31	
	3	3	32	
	4	4	33	
	5	5	34	
	6	1	35	
	7	2	27	
	8	4	28	
	9	5	19	
	10	1	13	
	11	2	12	
	12	3	13	
	13	4	11	
	14	5	12	
	15	1	13	
	16	2	14	
	NULL	NULL	NULL	

**If I had more time to improve on this project I would've designed my database project a bit more differently such as making the following changes:**

- "Email" and "Password" should both be of type VARCHAR, but "Email" should have a length of 255 and "Password" should have a length of 60
- Instead of using multiple tables for different types of mailbox folders (Inbox, Sent, Junk), consider using a single "Folder" table with a "Type" column that specifies the folder type (e.g. "Inbox," "Sent," "Junk"). This would allow you to store all messages in a single table and make it easier to query and manipulate the data