CSC 436 Assignment

Populating & Querying Your Database

Overview

In this assignment, you will navigate the full lifecycle of database management, starting with the setup and configuration of a new database in cPanel. With a focus on data integrity, you will learn to create tables with primary key, not null, unique, check, and foreign key constraints. Once your database is populated, you will dive into querying, mastering the art of retrieving, filtering, and aggregating data using SQL commands and operators. Finally, you will document your journey, detailing the steps taken to ensure data accuracy and consistency, and showcasing your database's structure, population, and insights extracted through SQL queries. Through this comprehensive experience, you will develop a robust understanding of database management principles and practices.

Part I: Setting Up Your Database

Before you start structuring your database and adding data, you'll need to access your cPanel account, create a new database, configure user access, and access the newly created database.

- ☑ ~~Complete the Logging Into cPanel section to log into your cPanel account. Please use the credentials emailed to you by your instructor.~~
- ☑ ~~Complete the Creating A Database section to create a new database.~~
- ☑ ~~Complete the Creating A User section to set yourself up as a user of the database you created.~~
- ☑ ~~Complete the Accessing The Database section to access the database you created.~~

⚠️ Your **cPanel accounts do not support real-time collaboration** for MySQL databases, so **one group member** must **create, populate, and query the database under their own account**.

All group members should actively participate by guiding and assisting that member during setup, including creating tables, importing data, and running queries. Engaging in this process **ensures everyone understands the database structure, tables, and data integrity constraints**. Once complete, the database will be exported and uploaded to your group's shared repository so all members can access and import it into their own cPanel accounts.

Part II: Populating Your Database

It's crucial to ensure that the tables you create adhere to specific requirements to maintain data integrity and consistency. Follow these procedures to create tables, insert values manually, and import databases and tables:

- **Creating Tables With Integrity Constraints:** When creating tables, ensure that each table includes appropriate integrity constraints to enforce data accuracy and consistency. These constraints may include:
  - Primary key constraints to uniquely identify each record in the table and prevent duplicate entries.
  - Not null constraints to prohibit null values for essential attributes.
  - Unique constraints to ensure attribute values are distinct within the table.
  - Check constraints to enforce specific conditions on attribute values.
  - Foreign key constraints to maintain referential integrity between related tables.

- **Inserting Data With Validations:** When inserting data manually, validate that the values being inserted adhere to the defined integrity constraints. Ensure that:
  - Mandatory attributes are provided with non-null values.
  - Unique attribute values do not conflict with existing data.
  - Attribute values meet any specified conditions enforced by check constraints.
  - Primary key values are unique for each record in the table.
  - Foreign key constraints are satisfied by referencing existing values in related tables.

- **Importing Databases With Consistency:** If importing databases or tables, verify that the imported data conforms to the integrity constraints defined for the target database. Perform data validation checks to ensure:
  - Imported data meets the requirements specified by integrity constraints.
  - Integrity constraints are enforced consistently across the imported tables.
  - Primary key values are unique and consistent across the imported data.
  - Any foreign key relationships are maintained and referenced correctly.

Review the following to ensure that your database is populated accurately and consistently:

☑ ~~Review the Creating A Table section to design and create tables with appropriate integrity constraints.~~

☑ ~~Review the Inserting Data section to insert values into your tables while adhering to defined integrity constraints.~~

□ Review the Importing A Database section to import external data sources into your database while ensuring data consistency and integrity.

☑ ~~Review the Viewing A Database section to explore your populated database and verify data accuracy and completeness.~~

Part III: Querying Your Database

Now that your database is populated with relevant data, it's time to harness the power of SQL to retrieve and manipulate that data. You'll perform queries on single and multiple relations using various SQL commands and operators. Follow these steps to effectively query your database:

- **Single-Relation Queries**
  - Use the **SELECT** clause to retrieve specific columns or all columns from a single table.
  - Use the **FROM** clause to specify the table from which you want to retrieve data.
  - Apply the **WHERE** clause with comparison and logical operators to filter rows based on specific conditions.
  - Employ arithmetic operators (+, -, *, /) to perform calculations within your queries.
  - Use the **ORDER BY** clause to sort query results based on specified column(s).

- **Multiple-Relation Queries**
  - Perform joins (i.e., **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN**, **FULL OUTER JOIN**) to combine data from multiple tables based on related columns.
  - Use the **NATURAL JOIN** to join tables based on columns with the same name.
  - Apply aggregate functions (i.e., **SUM**, **AVG**, **COUNT**, **MIN**, **MAX**) to calculate summary statistics across groups of data.
  - Use the **GROUP BY** clause to group query results based on one or more columns.
  - Apply aggregate functions alongside **GROUP BY** to perform group-level calculations.
  - Use the **HAVING** clause to filter group-level results based on aggregate function results.

- **Executing Queries**
  - Write SQL queries that leverage the above-mentioned commands and operators to extract meaningful insights from your database.
  - Test your queries in a SQL environment.
  - Verify the accuracy and relevance of query results by cross-referencing them with the expected outcomes.

Review the following to perform queries on your database:

☑ ~~Review the SQL Statements section to familiarize with common SQL statements to effectively communicate with your database.~~

☑ ~~Review the Querying The Database section to write queries and ask questions about the data.~~

Part IV: Documentation

For each entity in our database, we defined a primary key to uniquely identify every record. Each attribute is assigned an appropriate data type, such as INT(11), VARCHAR(50), or DECIMAL(10,2), to ensure proper data storage and validation.

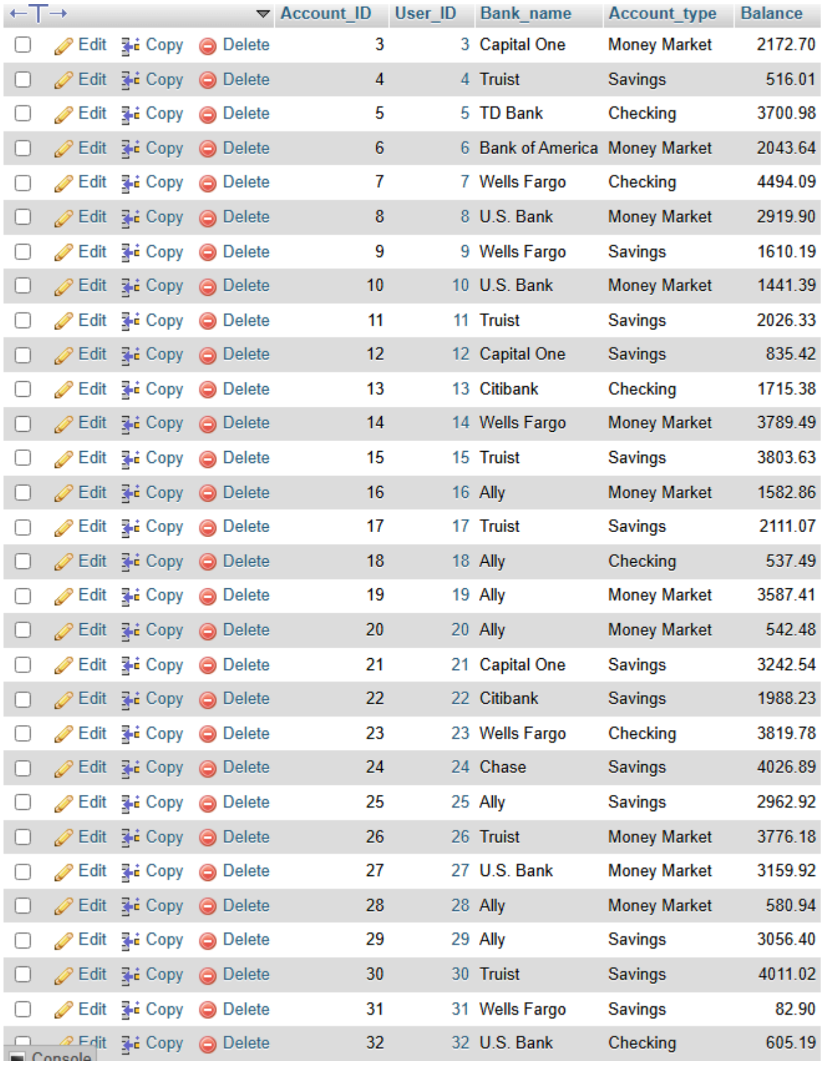
The only attribute that does not include a NOT NULL constraint is the description field in the Transaction table. All other attributes are marked as NOT NULL, as they are required for the database to function correctly and maintain data integrity.

We also implemented UNIQUE constraints on specific attributes to prevent duplicate values. These include: user_ID, account_ID, transaction_ID, category_ID, pay_ID, email, and phone_number.

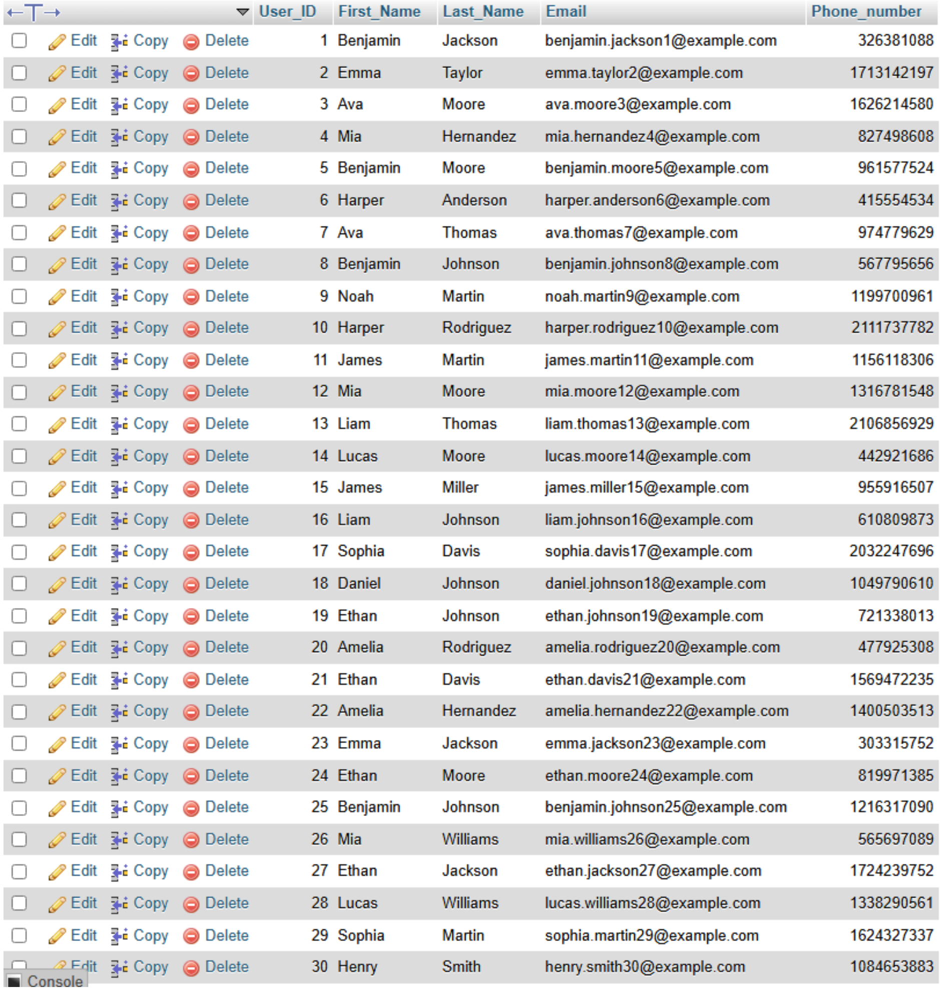
In addition, we established domain constraints to restrict certain attributes to predefined values. For example, account_type is limited to “checking” or “saving,” and transaction_type is limited to “expense” or “income.” These constraints help maintain consistency and ensure valid data entry throughout the system.

We have foreign keys in almost every entity table, unlike our original ER diagram. We found it was best to add foreign keys so that we were able to create the relationships needed. This will allow our tables to “interact” with each other with certain attributes being shared throughout the database.

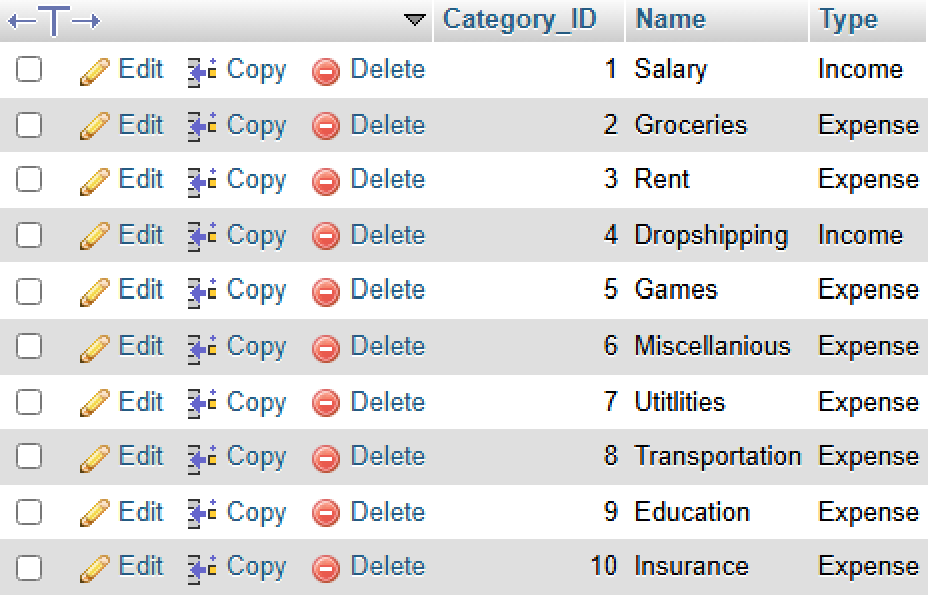
1. Include screenshots of EACH table in your database. You can use the **Browse** tab and take a screenshot of the data that is displayed. This will demonstrate that you have successfully populated the database.



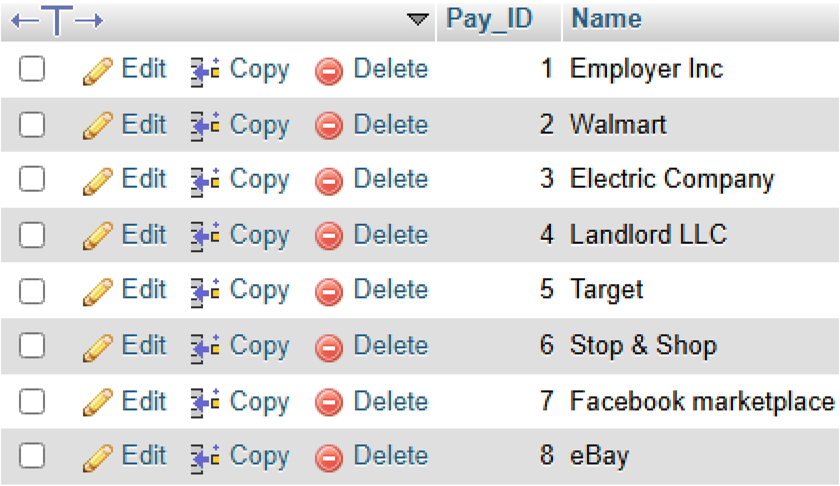
←T→				Account_ID	User_ID	Bank_name	Account_type	Balance
□	Edit	Copy	Delete	3	3	Capital One	Money Market	2172.70
□	Edit	Copy	Delete	4	4	Truist	Savings	516.01
□	Edit	Copy	Delete	5	5	TD Bank	Checking	3700.98
□	Edit	Copy	Delete	6	6	Bank of America	Money Market	2043.64
□	Edit	Copy	Delete	7	7	Wells Fargo	Checking	4494.09
□	Edit	Copy	Delete	8	8	U.S. Bank	Money Market	2919.90
□	Edit	Copy	Delete	9	9	Wells Fargo	Savings	1610.19
□	Edit	Copy	Delete	10	10	U.S. Bank	Money Market	1441.39
□	Edit	Copy	Delete	11	11	Truist	Savings	2026.33
□	Edit	Copy	Delete	12	12	Capital One	Savings	835.42
□	Edit	Copy	Delete	13	13	Citibank	Checking	1715.38
□	Edit	Copy	Delete	14	14	Wells Fargo	Money Market	3789.49
□	Edit	Copy	Delete	15	15	Truist	Savings	3803.63
□	Edit	Copy	Delete	16	16	Ally	Money Market	1582.86
□	Edit	Copy	Delete	17	17	Truist	Savings	2111.07
□	Edit	Copy	Delete	18	18	Ally	Checking	537.49
□	Edit	Copy	Delete	19	19	Ally	Money Market	3587.41
□	Edit	Copy	Delete	20	20	Ally	Money Market	542.48
□	Edit	Copy	Delete	21	21	Capital One	Savings	3242.54
□	Edit	Copy	Delete	22	22	Citibank	Savings	1988.23
□	Edit	Copy	Delete	23	23	Wells Fargo	Checking	3819.78
□	Edit	Copy	Delete	24	24	Chase	Savings	4026.89
□	Edit	Copy	Delete	25	25	Ally	Savings	2962.92
□	Edit	Copy	Delete	26	26	Truist	Money Market	3776.18
□	Edit	Copy	Delete	27	27	U.S. Bank	Money Market	3159.92
□	Edit	Copy	Delete	28	28	Ally	Money Market	580.94
□	Edit	Copy	Delete	29	29	Ally	Savings	3056.40
□	Edit	Copy	Delete	30	30	Truist	Savings	4011.02
□	Edit	Copy	Delete	31	31	Wells Fargo	Savings	82.90
□	Edit	Copy	Delete	32	32	U.S. Bank	Checking	605.19



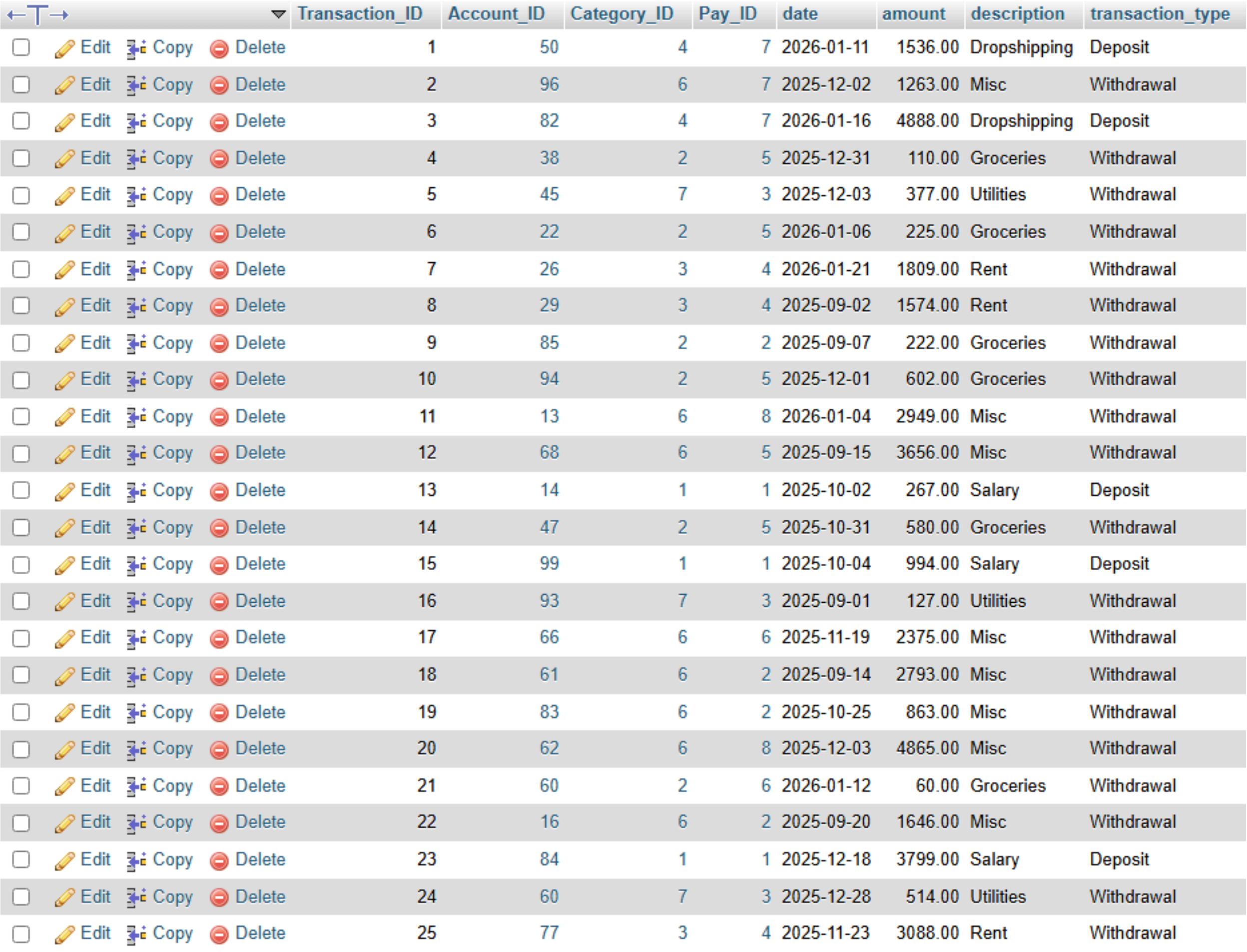
←T→	User_ID	First_Name	Last_Name	Email	Phone_number
□ Edit Copy Delete	1	Benjamin	Jackson	benjamin.jackson1@example.com	326381088
□ Edit Copy Delete	2	Emma	Taylor	emma.taylor2@example.com	1713142197
□ Edit Copy Delete	3	Ava	Moore	ava.moore3@example.com	1626214580
□ Edit Copy Delete	4	Mia	Hernandez	mia.hernandez4@example.com	827498608
□ Edit Copy Delete	5	Benjamin	Moore	benjamin.moore5@example.com	961577524
□ Edit Copy Delete	6	Harper	Anderson	harper.anderson6@example.com	415554534
□ Edit Copy Delete	7	Ava	Thomas	ava.thomas7@example.com	974779629
□ Edit Copy Delete	8	Benjamin	Johnson	benjamin.johnson8@example.com	567795656
□ Edit Copy Delete	9	Noah	Martin	noah.martin9@example.com	1199700961
□ Edit Copy Delete	10	Harper	Rodriguez	harper.rodriguez10@example.com	2111737782
□ Edit Copy Delete	11	James	Martin	james.martin11@example.com	1156118306
□ Edit Copy Delete	12	Mia	Moore	mia.moore12@example.com	1316781548
□ Edit Copy Delete	13	Liam	Thomas	liam.thomas13@example.com	2106856929
□ Edit Copy Delete	14	Lucas	Moore	lucas.moore14@example.com	442921686
□ Edit Copy Delete	15	James	Miller	james.miller15@example.com	955916507
□ Edit Copy Delete	16	Liam	Johnson	liam.johnson16@example.com	610809873
□ Edit Copy Delete	17	Sophia	Davis	sophia.davis17@example.com	2032247696
□ Edit Copy Delete	18	Daniel	Johnson	daniel.johnson18@example.com	1049790610
□ Edit Copy Delete	19	Ethan	Johnson	ethan.johnson19@example.com	721338013
□ Edit Copy Delete	20	Amelia	Rodriguez	amelia.rodriguez20@example.com	477925308
□ Edit Copy Delete	21	Ethan	Davis	ethan.davis21@example.com	1569472235
□ Edit Copy Delete	22	Amelia	Hernandez	amelia.hernandez22@example.com	1400503513
□ Edit Copy Delete	23	Emma	Jackson	emma.jackson23@example.com	303315752
□ Edit Copy Delete	24	Ethan	Moore	ethan.moore24@example.com	819971385
□ Edit Copy Delete	25	Benjamin	Johnson	benjamin.johnson25@example.com	1216317090
□ Edit Copy Delete	26	Mia	Williams	mia.williams26@example.com	565697089
□ Edit Copy Delete	27	Ethan	Jackson	ethan.jackson27@example.com	1724239752
□ Edit Copy Delete	28	Lucas	Williams	lucas.williams28@example.com	1338290561
□ Edit Copy Delete	29	Sophia	Martin	sophia.martin29@example.com	1624327337
□ Edit Copy Delete	30	Henry	Smith	henry.smith30@example.com	1084653883



←T→	Category_ID	Name	Type
□ Edit Copy Delete	1	Salary	Income
□ Edit Copy Delete	2	Groceries	Expense
□ Edit Copy Delete	3	Rent	Expense
□ Edit Copy Delete	4	Dropshipping	Income
□ Edit Copy Delete	5	Games	Expense
□ Edit Copy Delete	6	Miscellanious	Expense
□ Edit Copy Delete	7	Utitlities	Expense
□ Edit Copy Delete	8	Transportation	Expense
□ Edit Copy Delete	9	Education	Expense
□ Edit Copy Delete	10	Insurance	Expense



←T→ ▼	Pay_ID	Name
□ Edit Copy Delete	1	Employer Inc
□ Edit Copy Delete	2	Walmart
□ Edit Copy Delete	3	Electric Company
□ Edit Copy Delete	4	Landlord LLC
□ Edit Copy Delete	5	Target
□ Edit Copy Delete	6	Stop & Shop
□ Edit Copy Delete	7	Facebook marketplace
□ Edit Copy Delete	8	eBay

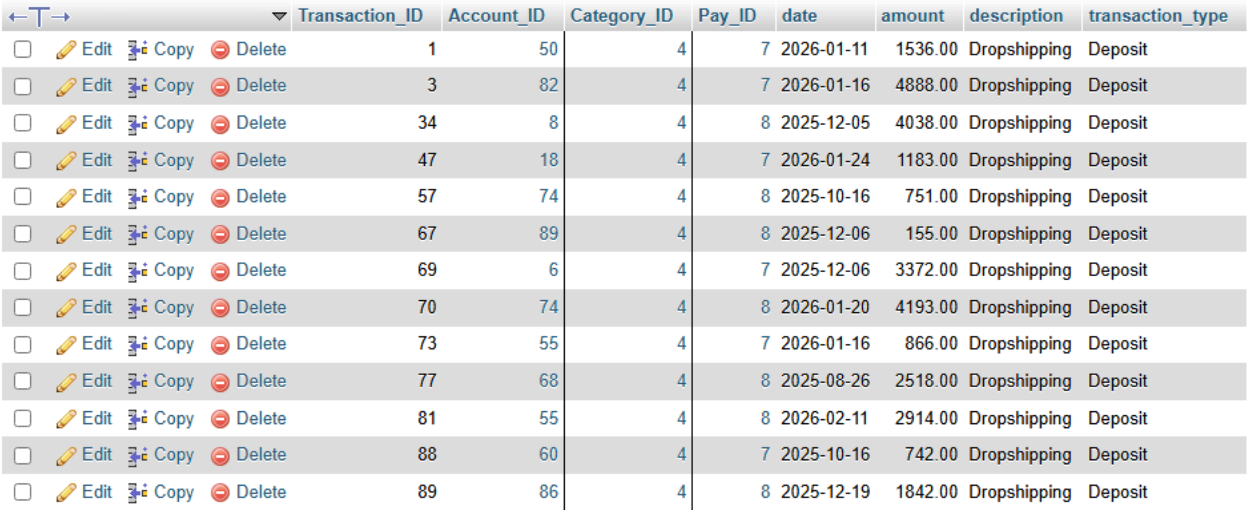


←T→ ▼	Transaction_ID	Account_ID	Category_ID	Pay_ID	date	amount	description	transaction_type
□ Edit Copy Delete	1	50	4	7	2026-01-11	1536.00	Dropshipping	Deposit
□ Edit Copy Delete	2	96	6	7	2025-12-02	1263.00	Misc	Withdrawal
□ Edit Copy Delete	3	82	4	7	2026-01-16	4888.00	Dropshipping	Deposit
□ Edit Copy Delete	4	38	2	5	2025-12-31	110.00	Groceries	Withdrawal
□ Edit Copy Delete	5	45	7	3	2025-12-03	377.00	Utilities	Withdrawal
□ Edit Copy Delete	6	22	2	5	2026-01-06	225.00	Groceries	Withdrawal
□ Edit Copy Delete	7	26	3	4	2026-01-21	1809.00	Rent	Withdrawal
□ Edit Copy Delete	8	29	3	4	2025-09-02	1574.00	Rent	Withdrawal
□ Edit Copy Delete	9	85	2	2	2025-09-07	222.00	Groceries	Withdrawal
□ Edit Copy Delete	10	94	2	5	2025-12-01	602.00	Groceries	Withdrawal
□ Edit Copy Delete	11	13	6	8	2026-01-04	2949.00	Misc	Withdrawal
□ Edit Copy Delete	12	68	6	5	2025-09-15	3656.00	Misc	Withdrawal
□ Edit Copy Delete	13	14	1	1	2025-10-02	267.00	Salary	Deposit
□ Edit Copy Delete	14	47	2	5	2025-10-31	580.00	Groceries	Withdrawal
□ Edit Copy Delete	15	99	1	1	2025-10-04	994.00	Salary	Deposit
□ Edit Copy Delete	16	93	7	3	2025-09-01	127.00	Utilities	Withdrawal
□ Edit Copy Delete	17	66	6	6	2025-11-19	2375.00	Misc	Withdrawal
□ Edit Copy Delete	18	61	6	2	2025-09-14	2793.00	Misc	Withdrawal
□ Edit Copy Delete	19	83	6	2	2025-10-25	863.00	Misc	Withdrawal
□ Edit Copy Delete	20	62	6	8	2025-12-03	4865.00	Misc	Withdrawal
□ Edit Copy Delete	21	60	2	6	2026-01-12	60.00	Groceries	Withdrawal
□ Edit Copy Delete	22	16	6	2	2025-09-20	1646.00	Misc	Withdrawal
□ Edit Copy Delete	23	84	1	1	2025-12-18	3799.00	Salary	Deposit
□ Edit Copy Delete	24	60	7	3	2025-12-28	514.00	Utilities	Withdrawal
□ Edit Copy Delete	25	77	3	4	2025-11-23	3088.00	Rent	Withdrawal

2. Five SQL queries were written to extract meaningful insights from the database. Each query serves a distinct purpose and uses different SQL clauses and operators including:

   ○ A query leveraging SELECT, FROM, and WHERE clauses with comparison and logical operators to filter rows based on specific conditions.

SELECT * FROM Transaction WHERE Category_ID = 4

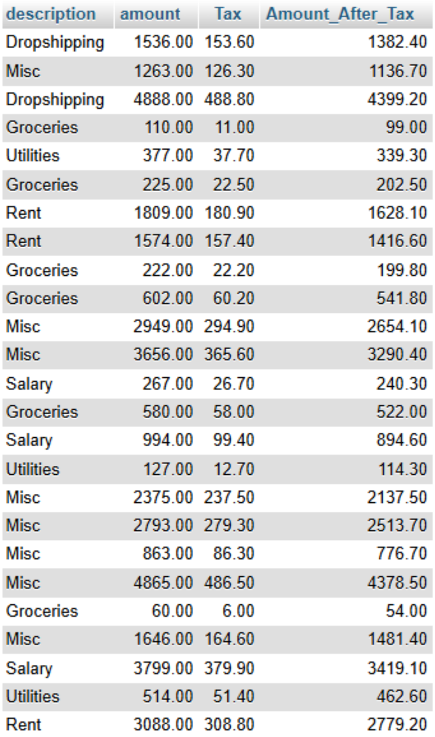


←T→			▼	Transaction_ID	Account_ID	Category_ID	Pay_ID	date	amount	description	transaction_type
□	Edit	Copy	Delete	1	50	4	7	2026-01-11	1536.00	Dropshipping	Deposit
□	Edit	Copy	Delete	3	82	4	7	2026-01-16	4888.00	Dropshipping	Deposit
□	Edit	Copy	Delete	34	8	4	8	2025-12-05	4038.00	Dropshipping	Deposit
□	Edit	Copy	Delete	47	18	4	7	2026-01-24	1183.00	Dropshipping	Deposit
□	Edit	Copy	Delete	57	74	4	8	2025-10-16	751.00	Dropshipping	Deposit
□	Edit	Copy	Delete	67	89	4	8	2025-12-06	155.00	Dropshipping	Deposit
□	Edit	Copy	Delete	69	6	4	7	2025-12-06	3372.00	Dropshipping	Deposit
□	Edit	Copy	Delete	70	74	4	8	2026-01-20	4193.00	Dropshipping	Deposit
□	Edit	Copy	Delete	73	55	4	7	2026-01-16	866.00	Dropshipping	Deposit
□	Edit	Copy	Delete	77	68	4	8	2025-08-26	2518.00	Dropshipping	Deposit
□	Edit	Copy	Delete	81	55	4	8	2026-02-11	2914.00	Dropshipping	Deposit
□	Edit	Copy	Delete	88	60	4	7	2025-10-16	742.00	Dropshipping	Deposit
□	Edit	Copy	Delete	89	86	4	8	2025-12-19	1842.00	Dropshipping	Deposit

This query's purpose is to show all the transactions that are in category 4, which is dropshipping

- A query employing arithmetic operators (+, -, *, /) to perform calculations within the queries.

```
SELECT description, amount,

  ROUND(amount * 0.10, 2)AS Tax,

  ROUND(amount - (amount * 0.10),2) AS Amount_After_Tax

FROM Transactions;
```



description	amount	Tax	Amount_After_Tax
Dropshipping	1536.00	153.60	1382.40
Misc	1263.00	126.30	1136.70
Dropshipping	4888.00	488.80	4399.20
Groceries	110.00	11.00	99.00
Utilities	377.00	37.70	339.30
Groceries	225.00	22.50	202.50
Rent	1809.00	180.90	1628.10
Rent	1574.00	157.40	1416.60
Groceries	222.00	22.20	199.80
Groceries	602.00	60.20	541.80
Misc	2949.00	294.90	2654.10
Misc	3656.00	365.60	3290.40
Salary	267.00	26.70	240.30
Groceries	580.00	58.00	522.00
Salary	994.00	99.40	894.60
Utilities	127.00	12.70	114.30
Misc	2375.00	237.50	2137.50
Misc	2793.00	279.30	2513.70
Misc	863.00	86.30	776.70
Misc	4865.00	486.50	4378.50
Groceries	60.00	6.00	54.00
Misc	1646.00	164.60	1481.40
Salary	3799.00	379.90	3419.10
Utilities	514.00	51.40	462.60
Rent	3088.00	308.80	2779.20

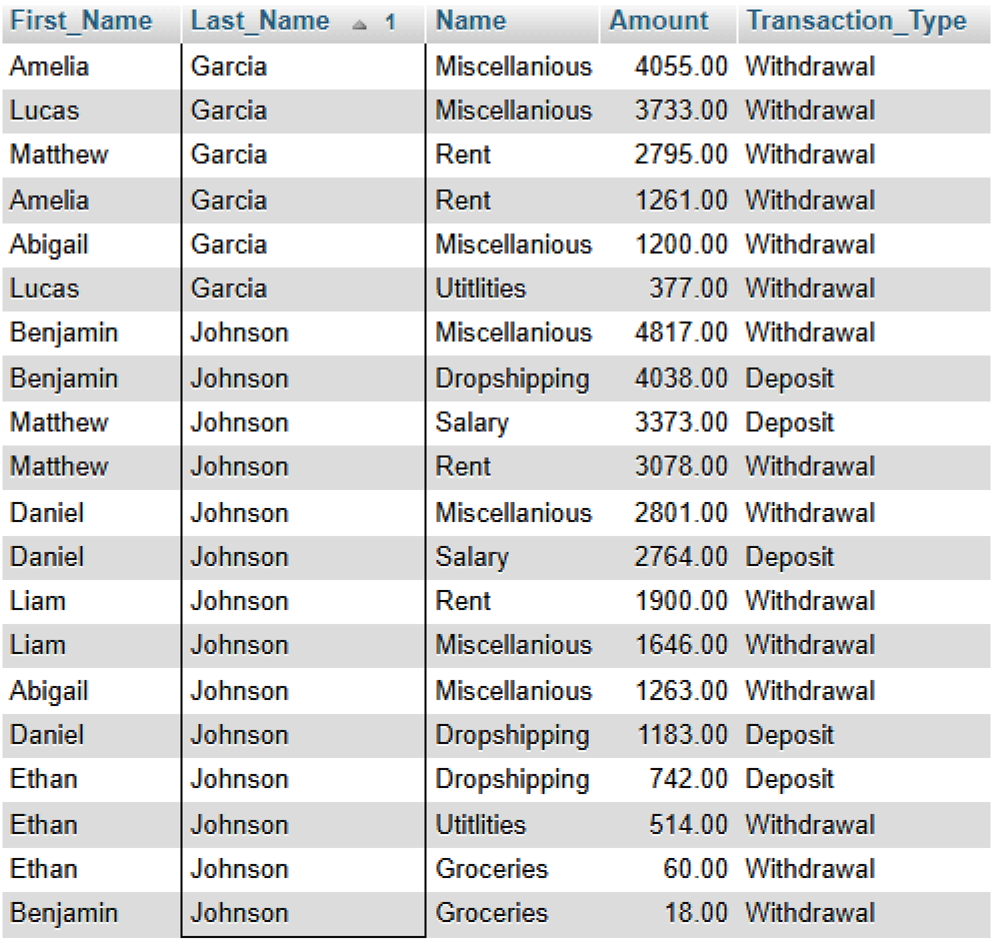
The purpose of this query is to show text calculations for each transaction. So in this case, just 10% of each transaction. One tab (Tax) is going to show the amount taken out, and the other(Amount_After_Tax) is going to show the new amount. (The ROUND function is just to make the results show 100.00 instead of 100.0000)

- ○ A query performing join expressions to combine data from multiple tables.

```
SELECT

  Account_Holder.First_Name,

  Account_Holder.Last_Name,

  Category.Name,

  Transactions.Amount,

  Transactions.Transaction_Type

FROM Transactions

JOIN Account ON Transactions.Account_ID = Account.Account_ID

JOIN Account_Holder Account_Holder ON Account.User_ID = Account_Holder.User_ID

JOIN Category ON Transactions.Category_ID = Category.Category_ID
```

WHERE Account_Holder.Last_Name IN ('Johnson','Garcia')

ORDER BY Account_Holder.Last_Name, Transactions.Amount DESC;



First_Name	Last_Name ▲ 1	Name	Amount	Transaction_Type
Amelia	Garcia	Miscellanious	4055.00	Withdrawal
Lucas	Garcia	Miscellanious	3733.00	Withdrawal
Matthew	Garcia	Rent	2795.00	Withdrawal
Amelia	Garcia	Rent	1261.00	Withdrawal
Abigail	Garcia	Miscellanious	1200.00	Withdrawal
Lucas	Garcia	Utitlities	377.00	Withdrawal
Benjamin	Johnson	Miscellanious	4817.00	Withdrawal
Benjamin	Johnson	Dropshipping	4038.00	Deposit
Matthew	Johnson	Salary	3373.00	Deposit
Matthew	Johnson	Rent	3078.00	Withdrawal
Daniel	Johnson	Miscellanious	2801.00	Withdrawal
Daniel	Johnson	Salary	2764.00	Deposit
Liam	Johnson	Rent	1900.00	Withdrawal
Liam	Johnson	Miscellanious	1646.00	Withdrawal
Abigail	Johnson	Miscellanious	1263.00	Withdrawal
Daniel	Johnson	Dropshipping	1183.00	Deposit
Ethan	Johnson	Dropshipping	742.00	Deposit
Ethan	Johnson	Utitlities	514.00	Withdrawal
Ethan	Johnson	Groceries	60.00	Withdrawal
Benjamin	Johnson	Groceries	18.00	Withdrawal

The query's intended purpose is to find all of the transactions that come from an account holder with the last name "Garcia" and "Johnson". It also organizes it by the descending amount of money.

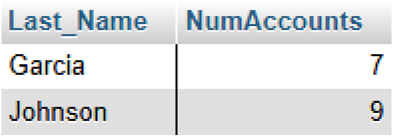
- ○ A query using the GROUP BY clause and applying aggregate functions to perform group-level calculations.

SELECT Last_Name, COUNT(*) AS NumAccounts

FROM Account_Holder

WHERE Last_Name IN ("Johnson","Garcia")

GROUP BY Last_Name;



Last_Name	NumAccounts
Garcia	7
Johnson	9

This query's intended purpose is to add the amount of account holders with the last names "Garcia" and "Johnson", grouped by the two last names.

Submission

When you're finished, complete the following steps to submit your work:

☑ ~~Export your document with responses as a **PDF file AND save** it **inside** your **documentation** folder. Refer to the following for documentation on how to do this:~~

- Google Docs (*File → Download → PDF Document*)
- Microsoft Word (*File → Save As / Export → PDF*)
- Pages (*File → Export To → PDF*)

☑ ~~Inside your repo, create a folder named **db** and export your database as an **SQL file** into that folder. The file should include all SQL code used to create tables, insert data, and run other queries. Follow the steps in the Exporting A Database section to export your database into a single SQL file.~~

☑ ~~All group members should then be able to pull this SQL file from the repo and upload it to their own cPanel accounts (see Importing A Database section).~~

☐ Upload all your changes to GitHub.

☐ **If you're using GitHub Desktop (GUI)**, complete the Uploading Changes (GitHub Desktop) section to upload your changes from your local device to GitHub.

☐ **If you're using Git (CLI)**, complete the Uploading Changes (GitHub CLI) section to upload your changes from your local device to GitHub.

ONE group member must paste the URL of your GitHub repository in the provided textbox in Brightspace. Click the blue *Submit* button to successfully submit your work for this assignment.

Grading Rubric

You can refer to the **Populating & Querying Your Database grading rubric** given in Brightspace for this assignment to find details on how your submission will be graded.