Database Case Study

Ashleigh Moore

IST 423 Spring 2024

Table of Contents

Background

This database project was based on the fictional California-based bookstore chain, Beyond Borders Books. The project involved creating a database that would track every aspect of the business. The intention was to focus on the main business aspects of books, sales, café, and staff, and have the option/ability to expand the database to meet business needs in the future. The database needed to include views for important business functions, including:

- Tracking inventory, both for individual locations and the entire company, as well as the ability to track the inventory of individual products.
- Tracking sales for each location and the company as whole, as well as tracking sales of specific items.
- Scheduling employees.
- Tracking how many hours each employee has worked over a given period of time, as well as calculating their pay over that time period.
- Monitoring staffing costs at each location.
- Calculating the profits of each location over a given period of time.

Implementing these views will provide a good base upon which to build additional views as needed.

Several business rules also needed to be implemented throughout the creation of this database. These business rules include one staff per sales transaction, a maximum limit of 10 of any one item purchased at a time, and a minimum wage set at $15 per hour.

Project Scope/Parts

The plan for this project began with defining the scope of the project and what would need to be done in order to complete it. After this the plan was to create an ER diagram that would act as the basis for the development. The plan for the development was to begin by writing create table statements for the entities and attributes outlined in the ER diagram. Once the tables were created, data would be generated and imported into the tables, and view queries would be written to fulfill the necessary requirements.

The plan for the execution of these processes was to begin with thoroughly reading the scenario documents off of which the project scope and requirements could be defined. The plan for creating the ER diagram was to begin by doing a rough sketch on paper, and then improving and refining that sketch once or twice, still on paper. After a decent sketch of an ER diagram was created, the online tool ERDPlus would then be used to create a neater and more polished diagram. Once the diagram was done, the plan was to start writing out the create table statements and defining attribute datatypes, as well as any necessary constraints for each table. Data would then be created from an online data generator which could be imported and used to test everything. After this, views would be created to retrieve data relevant to the different business aspects.

As for the actual completion of the project, the process began with defining the project scope which was largely completed with the Project Definition Documentation assignment. After establishing this definition, an ER diagram sketch was done on paper then improved upon and revised, as was the plan. The diagram was also created in ERDPlus as the plan intended. Once the ER diagram was established, it became clear that writing the statements to create the tables would be easier if first done in a Word document instead of going directly to pgAdmin 4. This

allowed the statements to be written all at once and allowed for easy revisions as needed. Datatypes for fields were also defined during this step, which could also be easily adjusted when necessary. Once all of the code for the tables was written in this document, the process of generating data to fill the tables began. The program used to generate the data was Mockaroo. Note that for the generation of most of the fields that needed a date (i.e., transaction dates, schedule dates, shift dates) was restricted to the month of March 2024 since there was minimal rows being generated, allowing for more specific views to be created and still see results. It is also worth noting that no data generators were found to have options for book titles, so they were substituted with movie titles for the purpose of the database testing, since that was the most similar category to the needed data. While going through this process of data generation, further revisions to datatypes and fields became apparent, and those were edited in the document with the code. Once fairly suitable data had been generated, everything was moved into pgAdmin 4. As the code for the tables was copied in and run, there were several errors, mostly minor syntax errors, which were then fixed in the current code, and pre-emptively fixed for the rest of the code. After all of the tables were created, the generated data was imported into them from csv files. The same process for writing the table creation code was then used for writing the view queries, which involved starting in a Word document, making changes throughout, and then copying it into pgAdmin. Once again, as queries were run, errors occurred, and revisions had to be made. Several times, the queries were successfully run, but upon reviewing the views, certain columns did not appear as was intended. The views were then dropped, revised, and tested again. Throughout the process of writing and running the statements and queries, a few resources were referenced for assistance, with the main one being the book Practical SQL (DeBarros, 2022).

Challenges

The only major challenge faced during the development of this database was the creation of the views associated with the time clock relation. The solution to this issue involved a lot of trial and error, and some research as well. There were also some minor issues faced, such as certain errors with the code, and some issues with the initial ER Diagram which had to be continuously revised.

Challenge 1

The first challenge faced was with the initial ER Diagram that was created, which can be seen on page 11 Appendix A. As development of the code to create the relations began, it was quickly realized that this ER Diagram was not thorough enough, in terms of the attributes for each entity, and the number of entities. The solution for this was to, as the tables were being created, make a note of the attributes and/or entities that were being added along the way, as well as a note of the things that might potentially need to be added that could be revisited later. After the tables were created, this list of additions and potential necessary additions was reviewed. Decisions were made about the potential necessary additions, and they were either added to the relations, or they were removed from the list. After this was complete, the ER Diagram was edited to match the tables and fields that had been created. The final ER Diagram can be seen on page 11 Appendix A. Each of the initial entities had its' attributes revised, and two new entities were added as well, the Schedule and Time Clock entities.

Challenge 2

The biggest challenge faced during this project was the creation of the views associated with the Time Clock Relation. The goal was to create views that could track employee hours and pay over a given time period, monitor staff costs for each location, and track profits for each location. After the addition of the time clock entity to the ER Diagram and the creation of the time clock table, this did not seem like it would be too difficult. An initial Staff Shifts View was created to show staff shifts and find the duration of this shifts, which was done by selecting clock_out – clock_in AS shift_duration, which when the code was run, seemed to have worked, but would cause issues later. The view to check how many hours a staff member had worked over a given time frame, using a week as an example, was then created based on the Staff Shifts View, which involved summing the durations of all the shifts worked by each staff throughout the week and displaying it as clocked_hours, which also seemed to work. The next step was to create a view that would calculate each staff's pay for the week. This seemed to be fairly simple, and it was thought that it could be done by joining the staff table to the previous view, and doing a select staff.hourly_pay multiplied by the clocked_hours for the week from the previous view. This, however, did not work, as it was realized that the shift duration and clocked hours had been created as date intervals, which could not be multiplied with a numeric datatype. The views in question were then dropped, and a cycle began of editing the view code, running it, getting errors or realizing it was still not what was needed, and repeating. After trying the known potential solutions, which did not work, research was done on how an interval datatype could be converted to a numeric datatype, on which the necessary calculations could be done. After a google search, the PostgreSQL documentation for date/time functions was consulted and a solution was found (PostgreSQL, n.d.). The solution was to do a select extract (epoch from clock_out – clock_in)

which extracted the number of seconds in the time interval, which was then divided by 60 to find the number of minutes in the time interval, or in other words the number of minutes of the shift duration. This was created as a numeric datatype, meaning it would work with the necessary calculations. In the next view created, the minutes were then divided by 60 again to find the number of hours of the shift duration. The remaining views were then created and worked with the code originally written. See pages 13-14 Appendix B for the final code used to create the views.

Challenge 3

A minor challenge faced throughout this project was syntax errors. The main ones being forgotten commas, spelling errors, misplaced parentheses, and incorrect apostrophes. These issues were fixed by reviewing the code to see if it was clear what the issue was, and if it was it was fixed and tried again. When it was not clear, the code contained in Practical SQL (DeBarros, 2022) was referenced to try and spot where the issue was.

Benefits

This database will help Beyond Borders Books maximize their efficiency and ability to reference important data by having all aspects of the company covered in one database. The company will now be able to effectively and efficiently do things such as:

- Track inventory of products.
- Monitor company sales numbers.
- Track company staffing costs.

- View profit margins for each location.
- Reduce stored data redundancy.
- More reliable data analysis and data-based decision-making.

Additionally, now that the core aspects of the business are covered within this database, it will be much easier to expand and add additional elements of the business. This could include those that are already a part of the business, such as music, movies, merch, and vendors, but also would easily allow for possible business expansions, such as additional locations, employees, or products.

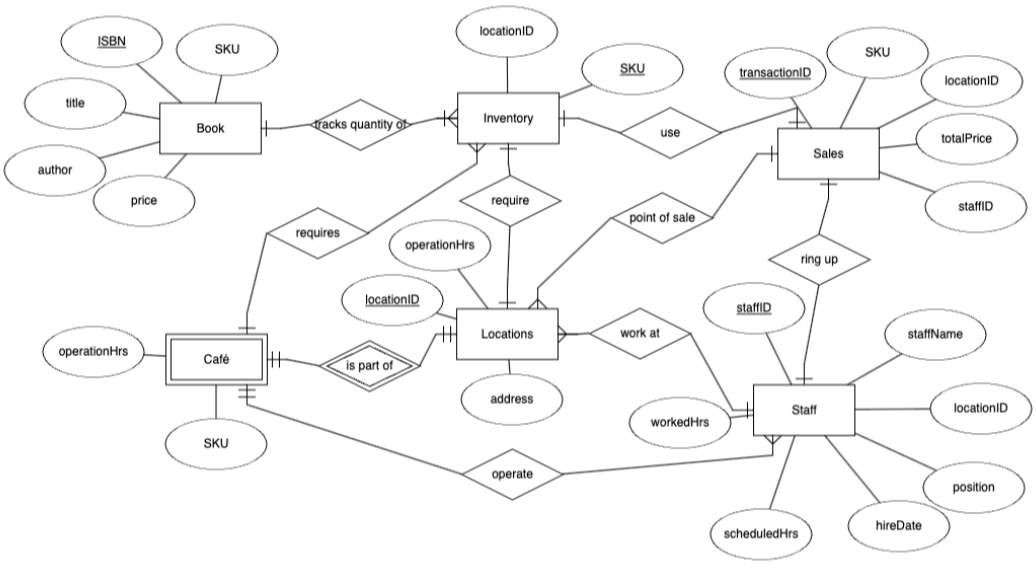
Citations

DeBarros, A. (2022). *Practical SQL, 2nd edition: A beginner's guide to storytelling with data*. No Starch Press.

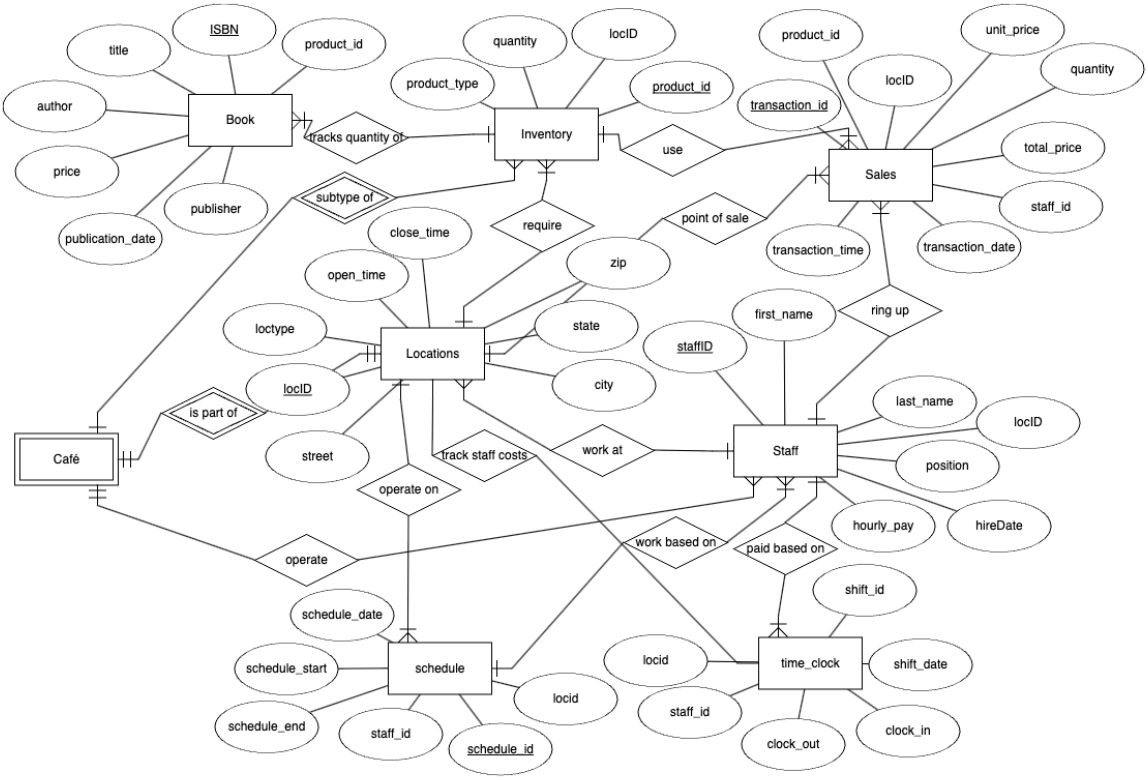
PostgreSQL. (n.d.). *9.9 Date/time functions and operators*. https://www.postgresql.org/docs/current/functions-datetime.html

Appendix A – ER Diagrams

Initial ER Diagram



Final ER Diagram



Appendix B – SQL Queries

Store Specific Inventory View Template

```
CREATE OR REPLACE VIEW locB001_inventory AS
      SELECT product_id, quantity, product_type
      FROM inventory
      WHERE locid = 'B001'
```

Note: locid B001 could be replaced with any location's locid to view that store's inventory. Change the name accordingly.

Inventory for Entire Company View

```
CREATE OR REPLACE VIEW company_inventory AS
      SELECT * FROM inventory
```

Sales Numbers for One Location for a Day View Template

```
CREATE OR REPLACE VIEW locB002_sales_3_20_2024 AS
      SELECT transaction_id, staff_id,
            transaction_time, product_id, total_price
      FROM sales
      WHERE transaction_date = '2024-03-20' AND locid = 'B002'
```

Note: To view sales for a different date, replace the transaction date in the WHERE clause with the desired date. To view sales for a different location, replace the locid in the WHERE clause with the locid of the desired location. Change the name of the view accordingly.

Sales Numbers for the Entire Company for a Day View Template

```
CREATE OR REPLACE VIEW company_sales_3_20_2024 AS
      SELECT transaction_id, staff_id,
```

```
        locid, transaction_time, product_id, total_price
    FROM sales
    WHERE transaction_date = '2024-03-20'
```

Note: To view sales for a different date, replace the transaction date in the WHERE clause with the desired date and change the name of the view accordingly.

Sales Numbers for a Specific Book View Template

```
CREATE OR REPLACE VIEW product_id_8_sales AS
    SELECT sales.transaction_id,
        sales.locid,
        sales.transaction_date,
        sales.quantity,
        book.product_id,
        book.title
    FROM sales
    JOIN book ON sales.product_id = book.product_id
    WHERE sales.product_id = '8'
```

Note: To view sales for a different product id, change the id in the WHERE clause. To view sales numbers for a book when given the title but not the product id, change the WHERE clause to be WHERE book.title = ‘[book title]’ .

Staff Shifts View

```
CREATE OR REPLACE VIEW staff_shifts AS
    SELECT staff_id, shift_date, clock_in, clock_out, locid,
        EXTRACT (EPOCH FROM clock_out - clock_in)/60 AS
            shift_duration_minutes
    FROM time_clock
```

Note: To view only shifts from a given time frame, add a WHERE shift_date <= '[date]' AND shift_date >= '[date]' clause at the end.

Staff Clocked Hours Over Given Time Frame View Template

```
CREATE OR REPLACE VIEW hours_week3_18 AS
      SELECT staff_id, locid,
            sum(shift_duration_minutes/60) AS clocked_hours
      FROM staff_shifts
      WHERE shift_date <= '3-24-2024' AND shift_date >= '3-18-2024'
      GROUP BY staff_id, locid
```

Note: To change the time period selected, replace the shift dates in the WHERE clause.

Staff Pay View Template

```
CREATE OR REPLACE VIEW staff_pay AS
      SELECT hours_week3_18.staff_id, hours_week3_18.locid,
            staff.hourly_pay,
            ROUND((staff.hourly_pay * clocked_hours)::numeric,2) AS
                  staff_week_pay
      FROM hours_week3_18
      JOIN staff ON hours_week3_18.staff_id = staff.staff_id
```

Note: This view's time frame is based on the Staff Clocked Hours Over Given Time Frame View, to change the time frame for this view, alter that view's time frame.

Specific Store Staff Costs View Template

```
CREATE OR REPLACE VIEW locB001_weekly_staff_costs AS
      SELECT locid,
            sum(staff_week_pay) AS total_costs
```

```
        FROM staff_pay
        WHERE locid = ‘B001’
```

Note: This view’s time frame is based on the Staff Clocked Hours Over Given Time Frame View, to change the time frame for this view, alter that view’s time frame. To view staff costs for a different store, replace the locid in the WHERE clause with the desired location.

Store Revenue View Template

```
CREATE OR REPLACE VIEW locB001_revenue_per_week AS
      SELECT locid,
            sum(total_price) AS locB001_revenue
      FROM sales
      WHERE locid = 'B001' AND transaction_date <= '3-24-2024' AND
            transaction_date >= '3-18-2024'
      GROUP BY locid
```

Note: To view revenue for a different location, change the locid in the WHERE clause to the desired location. To view revenue over a different time period, change the transaction dates in the WHERE clause to match the desired time frame.

Store Profits View Template

```
CREATE OR REPLACE VIEW locB001_profits AS
      SELECT locB001_revenue_per_week.locid,
            (locB001_revenue_per_week.locB001_revenue -
            locB001_weekly_staff_costs.total_costs) AS locB001_profits
      FROM locB001_revenue_per_week
      JOIN locB001_weekly_staff_costs ON locB001_revenue_per_week.locid
            = locB001_weekly_staff_costs.locid
```

Note: The time frame for this view is based on the time frame and location in the Store Revenue View, and the time frame in the Staff Clocked Hours Over Given Time Frame View.

Appendix C – Data Dictionary

Locations Relation

Field Name	locid
Field Type	Text (PK)
Field Description	Locid is the unique id of each location. Serves as a Primary Key for the Locations Relation.
References	Locations (PK) Inventory (FK) Staff (FK) Sales (FK) Schedule (FK) Time Clock (FK)

Field Name	loctype
Field Type	Text
Field Description	Loctype is the type of location.
References	Locations (PK)

Field Name	street
Field Type	Text
Field Description	Street refers to the street address of the location.
References	Locations (PK)

Field Name	city
Field Type	Text
Field Description	City is the city that the location is located in.
References	Locations (PK)

Field Name	state
Field Type	Text
Field Description	State is the U.S. state in which the location is located.
References	Locations (PK)

Field Name	zip
Field Type	Text
Field Description	Zip is the zipcode of the location's address.
References	Locations (PK)

Field Name	open_time
Field Type	Time
Field Description	Open_time is the time that the location opens every day.
References	Locations (PK)

Field Name	close_time
Field Type	Time
Field Description	Close_time is the time that the location closes every day.
References	Locations (PK)

Inventory Relation

Field Name	product_id
Field Type	Bigint (PK)
Field Description	Product_id is the unique id of each product. Serves as a Primary Key for the Inventory Relation. Must be unique.
References	Inventory (PK) Book (FK) Sales (FK)

Field Name	locid
Field Type	Text (FK)
Field Description	Locid is the unique id of the location in which each product is located. Serves as a Foreign Key for the Locations Relation
References	Inventory (FK) Locations (PK)

Field Name	quantity
Field Type	Integer
Field Description	Quantity is the amount in stock of each product.
References	Inventory (PK)

Field Name	product_type
Field Type	Text
Field Description	Product_type is the category of product that each product falls into.
References	Inventory (PK)

Book Relation

Field Name	title
Field Type	Text
Field Description	Title is the name of each book.
References	Book (PK)

Field Name	author
Field Type	Text
Field Description	Author is the author of each book.
References	Book (PK)

Field Name	publication_date
Field Type	Date
Field Description	Publication_date is the date that each book was published.
References	Book (PK)

Field Name	publisher
Field Type	Text
Field Description	Publisher is the entity that published the book.
References	Book (PK)

Field Name	isbn
Field Type	Varchar 20 (PK)
Field Description	Isbn is a unique identifier assigned to each book. It serves as a Primary Key for the Book Relation. Must be unique.
References	Book (PK)

Field Name	price
Field Type	Float
Field Description	Price is the cost for each book.
References	Book (PK)

Field Name	product_id
Field Type	Bigint (FK)
Field Description	Product_id is a unique identifier assigned to each product and therefore each book. It serves as a Foreign Key for the Inventory Relation.
References	Book (FK) Inventory (PK)

Staff Relation

Field Name	staff_id
Field Type	Bigserial (PK)
Field Description	Staff_id is the staff's unique identifier. Serves as a Primary Key for the Staff Relation. Must be unique
References	Staff (PK) Sales (FK) Schedule (FK) Time Clock (FK)

Field Name	first_name
Field Type	Text
Field Description	First_name is the staff member's first name.
References	Staff (PK)

Field Name	last_name
Field Type	Text
Field Description	Last_name is the staff member's last name.
References	Staff (PK)

Field Name	hire_date
Field Type	Date
Field Description	Hire_date refers to the date that the staff was officially hired.
References	Staff (PK)

Field Name	position
Field Type	Text
Field Description	Position is the main position/job title that the staff member holds.

References	Staff (PK)

Field Name	locid
Field Type	Text (FK)
Field Description	Locid is the unique identifier for the location at which the staff member is employed. Serves as a Foreign Key for the Locations Relation.
References	Staff (FK) Locations (PK)

Field Name	hourly_pay
Field Type	Float
Field Description	Hourly_pay is how much money the staff member earns in one hour of work. Minimum wage constraint: hourly_pay must be greater than or equal to 15.
References	Staff (PK)

Sales Relation

Field Name	transaction_id
Field Type	Bigserial (PK)
Field Description	Transaction_id is the unique identifier for each sales transaction that occurs. Serves as a Primary Key for the Sales Relation. Must be unique.
References	Sales (PK)

Field Name	staff_id
Field Type	Bigserial (FK)

Field Description	Staff_id is the unique identifier for the staff member responsible for each sales transaction. Serves as a Foreign Key for the Staff Relation.
References	Sales (FK) Staff (PK)

Field Name	locid
Field Type	Text (FK)
Field Description	Locid is the unique identifier for the location at which the transaction occurred. Serves as a Foreign Key for the Locations Relation
References	Sales (FK) Locations (PK)

Field Name	transaction_date
Field Type	Date
Field Description	Transaction_date is the date on which the sales transaction occurred.
References	Sales (PK)

Field Name	transaction_time
Field Type	Time
Field Description	Transaction_time is the time at which the sales transaction occurred.
References	Sales (PK)

Field Name	product_id
Field Type	Bigint (FK)
Field Description	Product_id is the unique identifier for the product which was purchased in the sales transaction. Serves as a Foreign Key for the Sales Relation
References	Sales (FK) Inventory (PK)

Field Name	unit_price
Field Type	Float
Field Description	Unit_price is the cost of one unit of the product being purchased.
References	Sales (PK)

Field Name	quantity
Field Type	Integer
Field Description	Quantity is the number of units purchased of the product being purchased.
References	Sales (PK)

Field Name	total_price
Field Type	Float
Field Description	Total_price is the cost of the entire sales transaction.
References	Sales (PK)

Schedule Relation

Field Name	schedule_id
Field Type	Bigint (PK)
Field Description	Schedule_id is the unique identifier assigned to each scheduled shift. Serves as a Primary Key for the Schedule Relation. Must be unique.
References	Schedule (PK)

Field Name	schedule_date
Field Type	Date
Field Description	Schedule_date is the date of the scheduled shift.
References	Schedule (PK)

Field Name	schedule_start
Field Type	Time
Field Description	Schedule_start is the start time of the scheduled shift.
References	Schedule (PK)

Field Name	schedule_end
Field Type	Time
Field Description	Schedule_end is the end time of the scheduled shift.
References	Schedule (PK)

Field Name	staff_id
Field Type	Bigserial (FK)
Field Description	Staff_id is the unique identifier for the staff intended to work the scheduled shift. Serves as a Foreign Key for the Staff Relation.
References	Schedule (FK) Staff (PK)

Field Name	locid
Field Type	Text (FK)
Field Description	Locid is the unique identifier for the location at which the shift is scheduled. Serves as a Foreign Key for the Locations Relation.
References	Schedule (FK) Locations (PK)

Time Clock Relation

Field Name	shift_id
Field Type	Bigint (PK)

Field Description	Shift_id is the unique identifier for the worked shift. Serves as a Primary Key for the Time Clock Relation. Must be unique.
References	Time Clock (PK)

Field Name	shift_date
Field Type	Date
Field Description	Shift_date is the date of the worked shift.
References	Time Clock (PK)

Field Name	clock_in
Field Type	Time
Field Description	Clock_in is the time that the staff member clocked in for work.
References	Time Clock (PK)

Field Name	clock_out
Field Type	Time
Field Description	Clock_out is the time that the staff member clocked out of work.
References	Time Clock (PK)

Field Name	staff_id
Field Type	Bigserial (FK)
Field Description	Staff_id is the unique identifier for the staff member who clocked in for each shift. Serves as a Foreign Key for the Staff Relation.
References	Time Clock (FK) Staff (PK)

Field Name	locid
Field Type	Text (FK)

Field Description	Locid is the unique identifier for the location at which the shift was worked. Serves as a Foreign Key for the Locations Relation.
References	Time Clock (FK) Locations (PK)