# 1. CREATE\_ACCOUNT\_PP (Louise)

This procedure creates a new account to the BC\_ACCOUNT table and returns the new identifier to created account. This procedure also checks if the required parameters is not NULL and making sure that there will be no duplicate emails if the new account has an email that exists.

# 2. PURCHASE\_MEMBERSHIP\_PP (Mariam)

This procedure adds a new membership to the BC\_MEMBERSHIP table. It gives an account and its membership attributes for the account. This procedure returns also the identifier for the account. Lastly, this procedure checks if the required parameters are not NULL and that the pass type, pass id, and the exists.

# 3. CREATE\_STATION\_PP (Mehmet)

This procedure adds a new station to the BC\_STATION table. It also checks if the latitude and longitude are correct values and that the capacity of number of available docks and vehicles does not exceed the station capacity. Lastly, it also checks if the given program exists.

# 4. CREATE\_BICYCLE\_PP (Abdul)

This procedure adds a new bicycle to the BC\_BICYCLE table. It creates a bicycle with the given attributes on the parameters and returns the new identifier of the created bicycle. This procedure also checks if the required parameters is not NULL and that the parameters are permitted by the CHECK constraint. Lastly, it checks if the power, latitude and longitude, and capacity values are correct.

# 5. ADD\_DOCK\_PP (Berat)

This procedure adds a new dock to the BC\_DOCK table. This procedure also checks if the required parameters is not NULL and that the station and bicycle exist. This procedure also checks if the dock status is correct in relation with the CHECK constraints and that the dock numbers are correct.

# 6. BEGIN\_TRIP\_PP (Mehmet)

This procedure adds a new trip to the BC\_TRIP table and records the bicycle, start station and the membership of the account. It updates the BC\_DOCK and BC\_STATION and changes the status for the dock and station in terms of availability. It also determines the start station automatically. It also updates the status for the bicycle, and returns the identifier for the new trip. This procedure also checks if the required parameters is not NULL. Lastly, it checks if the given membership and bicycle exists and that the bicycle is currently docked.

# 7. END\_TRIP\_PP (Abdul)

This procedure updates an existing trip to give the trip and end time, duration, distance, cost and an end station. This procedure does update the BC\_DOCK and BC\_STATION to update the changes for the status. It finds the identifier for the trip by using the ID for the bicycle. Checks to see if the dock is unoccupied and throws an error if it is occupied. It calculates the duration by using the difference between end time and start time. It also updates the station status if all the docks for the station is either out of service of occupied it will update the status to not accepting (0) and if there are some docks available to (1). If there are some bicycles available on the station it is set to (1) because it is renting bicycles. Additionally, this procedure checks if the required parameters are not NULL and that the bicycle exists. Lastly, it checks if the start time is earlier than end time, the dock is available.

# 8. UPDATE\_STATION\_STATUS\_PP (Mehmet)

This procedure updates the station status by querying the BC\_DOCK table to determine how many docks and bicycles on the given station are available. It also updates the flag if the station is renting or accepting returns. Lastly, the procedure checks if the required inputs is not NULL and if the given station exists.

# 9. UPDATE\_BICYCLE\_STATUS\_PP – Version 1 (Berat)

This procedure adds a new row to the BC\_BICYCLE\_STATUS table by reflecting the latitude, longitude, power, range and status of the bicycle. It also updates the BC\_BICYCLE table row for the given bicycle. In this table it updates the current status of the bicycle and its columns in this table. In addition, this procedure checks if the required inputs is not NULL, if the bicycle exists, and if the correct values in the constraints are given as a value to the columns.

# 10. UPDATE\_BICYCLE\_STATUS\_PP – Version 2 (Berat)

This procedure does exactly the same as Version 1, but it does not update the columns power and rage due to if the bicycle is not electric etc.

# 11. NUMBER\_OF\_BICYCLES\_PF (Mehmet)

This function returns the number of bicycles that are available for the given type for the given station id. This procedure also makes sure regards of the case of the input the bicycle type is written in, gives back the same result. If no type is written in the parameter, it returns a value of zero. Lastly, this procedure checks if the required values is not NULL.

# 12. REMOVE\_BICYCLE\_PP (Abdul)

This procedure takes a bicycle out of service so the bicycle cannot be booked, removing it from its dock and updates the status to unavailable. This procedure also calls for another procedure to update the value as NULL for the longitude and latitude columns for the given bicycle.

# 13. LIST\_AVAILABLE\_BICYCLES\_PP (Louise)

This procedure lists out all the available bicycles there is on the given station and the bicycle type by printing it out. If there are no bicycles available it prints that there are no bicycles available. Lastly, it checks if the required parameters is not NULL and if the given station ID exists.

# 14. BICYCLE\_SUMMARY\_PP (Mariam)

This procedure lists all the bicycles with its ID, type, number of trips and the total distance for a given timeframe. The bicycle needs to be used at least once through the timeframe for it to be on the list. It also inverts the dates if the start time is later than the end time so the procedure can run correctly. Lastly, the procedure checks if the required parameters is not NULL and that the program ID exists.