

## Własny skrypt generujące dane do bazy danych

W tym etapie stworzyłem własny skrypt, który generował rekordy do bazy. Dzięki temu moja baza posiadała pokaźną ilość rekordów, aby umożliwić mi testowanie zapytań. Całość skryptów tworzących rekordy dla danej tabeli dostępna w linku poniżej nazwy tabeli.

### **INSERT Contracts – 1909 rekordów**

[https://github.com/adrianpikul/projectDB/blob/main/INSERT%20Contracts%20-%201909 records.sql](https://github.com/adrianpikul/projectDB/blob/main/INSERT%20Contracts%20-%201909%20records.sql)

### **INSERT Customers – 1200 rekordów**

[https://github.com/adrianpikul/projectDB/blob/main/INSERT%20Customers%20-%201200 records.sql](https://github.com/adrianpikul/projectDB/blob/main/INSERT%20Customers%20-%201200%20records.sql)

### **INSERT DailyRaport – 15 917 rekordów**

[https://github.com/adrianpikul/projectDB/blob/main/INSERT%20DailyRaport-%2015917 records.sql](https://github.com/adrianpikul/projectDB/blob/main/INSERT%20DailyRaport-%2015917%20records.sql)

### **INSERT ProjFinance – 179 rekordów**

[https://github.com/adrianpikul/projectDB/blob/main/INSERT%20ProjFinance%20-%20179 records.sql](https://github.com/adrianpikul/projectDB/blob/main/INSERT%20ProjFinance%20-%20179%20records.sql)

### **INSERT Projects – 179 rekordów**

[https://github.com/adrianpikul/projectDB/blob/main/INSERT%20Projects-%20179 records.sql](https://github.com/adrianpikul/projectDB/blob/main/INSERT%20Projects-%20179%20records.sql)

### **INSERT Rules – 3 rekordy**

[https://github.com/adrianpikul/projectDB/blob/main/INSERT%20Rules%20-%203 records.sql](https://github.com/adrianpikul/projectDB/blob/main/INSERT%20Rules%20-%203%20records.sql)

### **INSERT Workers – 90 rekordów**

[https://github.com/adrianpikul/projectDB/blob/main/INSERT%20Workers%20-%2090 records.sql](https://github.com/adrianpikul/projectDB/blob/main/INSERT%20Workers%20-%2090%20records.sql)

```
INSERT INTO Rules(RuleAccess)
VALUES ('PRACOWNIK'), ('KSIEGOWA'), ('SZEFE');
```

Wycinek 1. Przykładowy kod do umieszczania danych w tabeli.

**Przykładowe zapytania SELECT w celu sprawdzenia poprawności danych.**

Wszystkie przykładowe zapytania dostępne pod adresem:

<https://github.com/adrianpikul/projectDB/blob/main/SELECTS.sql>

```
SELECT Name, Surname FROM projDB.Workers WHERE idWorkers = 2;
```

Wycinek 2 Pobierz imię i nazwisko pracownika.

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Name	Surname		
Kacper	Karpiński		

Rysunek 1 Rysunek przedstawia rezultat wykonania powyższego polecenia SELECT.

```
SELECT * FROM Projects WHERE Area > 1700 ORDER BY Area;
```

Wycinek 3 Pobierz wszystkie projekty, gdzie powierzchnia jest większa niż 1700.

Result Grid		Filter Rows		Export		Wrap Cell Contents			
idProjects	Name	DateStart	DateEnd	State	Comment	Area	idCustomers	idProjFinance	idManager
15	7qrLDopzMuG0sv5	2015-06-06	2015-06-20	Ukonczony	dolor	1736.59	939	15	80
90	H71KNUdhXqNYyI	2017-09-06	2017-09-20	Ukonczony	Quaerat	1740.77	843	90	33
44	pJcF65RXtdyKPY0	2016-05-01	2016-05-15	Ukonczony	autem	1764.76	489	44	51
97	w0EGk8sISIN2zK	2017-11-25	2017-12-07	Problem	minima.	1774.14	1115	97	17
96	uGjILNqoloJYmYP	2017-11-17	2017-11-25	Problem	adipiscing doloribus	1816.91	520	96	89
126	bdnVWwckPHC2o7qF	2018-10-07	2018-10-20	Problem	nesciunt minima.	1817.95	551	126	49
155	eWAttswrq3GPF22	2019-08-22	2019-09-01	Problem	tempore	1833.59	530	155	66
9	T9PVQaWKhXG3M7B	2015-03-27	2015-04-05	Ukonczony	harum, facilis	1838.81	944	9	23
135	u9Z4hdJmshOFSLW	2019-01-09	2019-01-21	Problem	consectetur	1841.94	986	135	59
1	tyshQRUjUzLTxcj	2015-01-01	2015-01-15	Problem	elit. aliquam	1858.63	1036	1	27
116	1jZosFJFmTXBM6W	2018-06-26	2018-07-07	Problem	facere libero	1858.63	1199	116	54
125	03znnVGA/nlwu31	2018-06-23	2018-10-07	Ukonczony	ah	1860.71	736	125	64

Rysunek 2 Rysunek przedstawia fragment rezultatu wykonania powyższego polecenia SELECT.

Result Grid   Filter Rows:   Export:   Wrap Cell Content:						
	idContracts	DateStart	DateEND	Position	RateDay	idWorkers
▶	1356	2015-01-26	2015-05-09	Customer Relationships	153.32	64
	1357	2015-05-09	2015-09-02	Customer Relationships	153.32	64
	1358	2015-09-02	2015-12-13	Customer Relationships	153.32	64
	1359	2015-12-13	2016-02-11	Customer Relationships	153.32	64
	1360	2016-02-11	2016-08-03	Customer Relationships	153.32	64
	1361	2016-08-03	2017-01-15	Customer Relationships	153.32	64
	1362	2017-01-15	2017-07-09	Customer Relationships	153.32	64
	1363	2017-07-09	2018-01-02	Customer Relationships	153.32	64
	1364	2018-01-02	2018-04-22	Customer Relationships	153.32	64

**Rysunek 3** Rysunek przedstawia fragment rezultatu wykonania powyższego polecenia SELECT.

```
SELECT * FROM DailyRaport WHERE idProjects = 88 ORDER BY idDailyRaport ;
```

**Wycinek 5** Pobierz wszystkie dzienne raporty dla projektu, sortuj według id raportu.

Result Grid   Filter Rows:   Export:   Wrap Cell Content:				
	idDailyRaport	Date	idWorkers	idProjects
▶	7981	2017-08-16	43	88
	7982	2017-08-16	10	88
	7983	2017-08-16	9	88
	7984	2017-08-16	51	88
	7985	2017-08-16	54	88
	7986	2017-08-17	62	88
	7987	2017-08-17	79	88
	7988	2017-08-17	50	88
	7989	2017-08-17	10	88

**Rysunek 4** Rysunek przedstawia fragment rezultatu wykonania powyższego polecenia SELECT.

```
SELECT * FROM mydb.DailyRaport WHERE Date BETWEEN
STR_TO_DATE('2,1,2019', '%d,%m,%Y') AND
STR_TO_DATE('18,1,2020', '%d,%m,%Y');
```

**Wycinek 6 Pobierz wszystkie dzienne raporty pomiędzy datami.**

	idDailyRaport	Date	idWorkers	idProjects
▶	11843	2019-01-02	42	134
	11844	2019-01-02	30	134
	11845	2019-01-02	47	134
	11846	2019-01-02	55	134
	11847	2019-01-03	62	134
	11848	2019-01-03	21	134
	11849	2019-01-03	11	134
	11850	2019-01-03	20	134
	11851	2019-01-04	36	134
	11852	2019-01-04	49	134
	11853	2019-01-04	26	134
	11854	2019-01-04	14	134

**Rysunek 5 Rysunek przedstawia fragment rezultatu wykonania powyższego polecenia SELECT.**

```
SELECT dWorkers, Name, Surname, Phone, Mail, Pesel FROM
mydb.Workers WHERE idWorkers =
(SELECT idManager FROM mydb.Projects WHERE idProjects = 34);
```

**Wycinek 7 Pobierz dane kierownika projektu.**

	idWorkers	Name	Surname	Phone	Mail	Pesel
▶	57	Joachim	Nowacki	942764374	joachimnowacki69@158843.pl	63062287506
*	NULL	NULL	NULL	NULL	NULL	NULL

**Rysunek 6 Rysunek przedstawia rezultat wykonania powyższego polecenia SELECT.**

	Name	DateStart	DateEnd	Budget
►	UpJ6W1DSZRku2KN	2015-01-02	2015-01-18	459701
	TE9vzRwk1rs552	2015-01-19	2015-03-13	381974
	k2GI6k14LikWRFi	2015-03-14	2015-04-21	222166
	lQBjLqUKt1aKY9p	2015-04-22	2015-05-24	142481
	38vHWFit2Kc6rbp	2015-05-25	2015-07-20	141391
	ZjdeEApi4NGBQ	2015-07-21	2015-08-15	405806
	Qs5BDbjqVYE9RG1	2015-08-16	2015-10-28	400865
	LxIMmpoEL7LVyZk	2015-10-29	2015-12-11	208770
	2njH0v3Urb9J56s	2015-12-12	2016-02-13	234213

**Rysunek 7** Rysunek przedstawia fragment rezultatu wykonania powyższego polecenia SELECT.

```
SELECT p.Name, c.CompanyName, c.NIP, c.City, c.Street,
c.ZipCode FROM projDB.Projects p
JOIN projDB.Customers c ON p.idProjects = c.idCustomers;
```

**Wycinek 9** Pobierz nazwę projektu i szczegółowe informacje o klientach.

	Name	CompanyName	NIP	City	Street	ZipCode
►	UpJ6W1DSZRku2KN	Michał Sikora512	47031054841	Złoczew	Juliusza Słowackiego 873	81-330
	TE9vzRwk1rs552	Lech Janiszewski488	24399422840	Łochów	Okopowa 163	37-936
	k2GI6k14LikWRFi	Julia Sobolewska517	22743102288	Moryń	Garncarska 433	11-397
	lQBjLqUKt1aKY9p	Rozalia Wrobel537	42609108945	Nowa Sarzyna	Sienna 84	69-685
	38vHWFit2Kc6rbp	Ada Kowalewska512	48007975568	Morąg	Toruńska 519	64-729
	ZjdeEApi4NGBQ	Patrycja Czerwinska218	66476274699	Wieluń	gen. Władysława Andersa 112	12-107
	Qs5BDbjqVYE9RG1	Joanna Czajkowska593	15367145337	Wojkowice	Łąkowa 378	97-610
	LxIMmpoEL7LVyZk	Zaneta Kołodziej597	89539911009	Ząbki	Poznańska 342	30-305
	2njH0v3Urb9J56s	Jan Urban41	28537326543	Leżajsk	Szewska 579	75-565
	QVZIPYO4RVZsC7	Olga Maj344	43502137298	Ostroróg	Bolesława Limanowskiego 318	56-917

**Rysunek 8** Rysunek przedstawia fragment rezultatu wykonania powyższego polecenia SELECT.

```
SELECT c.Name, c.Surname, c.CompanyName, c.NIP, p.Name FROM
mydb.Customers c
RIGHT JOIN mydb.Projects p ON p.idCustomers = c.idCustomers;
```

**Wycinek 10** Pobierz klientów i nazwę projektu.

	Name	Surname	CompanyName	NIP	Name
►	Mirosław	Laskowski	MirosławLaskowski293	72089119457	tyshQrUjUzLtXcj
	Marzena	Marciniak	MarzenaMarciniak518	84712732751	JrazypDdsda8U1B
	Witold	Kosiński	WitoldKosinski864	26067519484	qRqUR1ea1n4fhQC
	Paweł	Bednarz	PawełBednarz132	72404701105	4iG13GF6oaI2oGD
	Sylwia	Kamińska	SylwiaKaminska949	31636419596	V8Afj7hDmu31Rct
	Tadeusz	Jarosz	TadeuszJarosz561	64497333882	R2fxPWGj9M4biCu

**Rysunek 9** Rysunek przedstawia fragment rezultatu wykonania powyższego polecenia SELECT.

```
SELECT d.idDailyRaport, p.idProjects, p.Name, w.Surname,
w.Name FROM projDB.DailyRaport d
JOIN projDB.Projects p ON d.idProjects = p.idProjects
JOIN projDB.Workers w ON d.idWorkers = w.idWorkers
ORDER BY d.idDailyRaport, w.Surname, w.Name;
```

**Wycinek 11** Pobierz id raportu, nazwę projektu, imię i nazwisko pracowników.

	idDailyRaport	idProjects	Name	Surname	Name
►	1	2	TE9vzRwkl1rs552	Wolski	Hubert
	2	2	TE9vzRwkl1rs552	Kwiatkowska	Agnieszka
	3	2	TE9vzRwkl1rs552	Brzeziński	Andrzej
	4	2	TE9vzRwkl1rs552	Orłowski	Zenon
	5	2	TE9vzRwkl1rs552	Woźniak	Dorota
	6	2	TE9vzRwkl1rs552	Kurek	Izabela

**Rysunek 10** Rysunek przedstawia fragment rezultatu wykonania powyższego polecenia SELECT.

### Przykładowe funkcjonalności (procedury) bazy danych.

Wszystkie procedury są dostępne pod adresem:

<https://github.com/adrianpikul/projectDB/blob/main/PROCEDURES.sql>

```

DROP PROCEDURE IF EXISTS mydb.addprojFinanceOne ;
DELIMITER //
CREATE PROCEDURE mydb.addprojFinanceOne
(
    IN IBudget FLOAT,
    IN IExpectedProfit FLOAT,
    IN IBonusManager FLOAT,
    IN IPenaltyPerDay FLOAT,
    IN ICosts FLOAT,
    IN IAdditionalCosts FLOAT
)
BEGIN
    INSERT INTO mydb.ProjFinance(Budget, ExpectedProfit,
    BonusManager, PenaltyPerDay, Costs, AdditionalCosts)
    VALUES (IBudget, IExpectedProfit, IBonusManager,
    IPenaltyPerDay, ICosts, IAdditionalCosts);
END //
DELIMITER ;

CALL mydb.addprojFinanceOne
(187554.2,454332.3,5642.3,1.2,56254, 1643.4);

SELECT * FROM ProjFinance ORDER BY idProjFinance DESC;

```

**Wycinek 12 Dodanie nowego rekordu do tabeli ProjFinance.**

	idProjFinance	Budget	ExpectedProfit	BonusManager	PenaltyPerDay	Costs	AdditionalCosts
►	181	187554	454332	5642.3	1.2	56254	1643.4

**Rysunek 11 Rysunek przedstawia rezultat dla powyższej procedury.**

```

DROP PROCEDURE IF EXISTS mydb.getAllDRForProjBetweenDate ;
DELIMITER //
CREATE PROCEDURE getAllDRForProjBetweenDate
(
    IN STARTDATE VARCHAR(10),
    IN ENDDATE VARCHAR(10),
    IN IDPROJECT INT
)
BEGIN
    SELECT * FROM mydb.DailyRaport
    WHERE Date BETWEEN STR_TO_DATE(STARTDATE, '%d,%m,%Y')
    AND STR_TO_DATE(ENDDATE, '%d,%m,%Y')
    AND DailyRaport.idProjects= IDPROJECT
    ORDER BY Date;
END //
DELIMITER ;
CALL mydb.getAllDRForProjBetweenDate ('2,1,2015', '18,1,2022',
6);

```

**Wycinek 13** Procedura pobierająca wszystkie karty pracy w przedziale czasowym dla projektu.

Result Grid   Filter Rows:   Export:   Wrap Cell C				
	idDailyRaport	Date	idWorkers	idProjects
▶	510	2015-02-23	50	6
	511	2015-02-23	68	6
	512	2015-02-23	47	6
	513	2015-02-23	67	6
	514	2015-02-23	6	6
	515	2015-02-23	21	6
	516	2015-02-24	6	6

**Rysunek 12** Rysunek przedstawia fragment rezultatu dla powyższej procedury.



```

DROP PROCEDURE IF EXISTS mydb.getAllDRForWorkBetweenDate ;
DELIMITER //
CREATE PROCEDURE getAllDRForWorkBetweenDate
(
    IN STARTDATE VARCHAR(10),
    IN ENDDATE VARCHAR(10),
    IN IDWORKER INT
)
BEGIN
    SELECT * FROM DailyRaport
    WHERE Date BETWEEN STR_TO_DATE(STARTDATE, '%d,%m,%Y')
    AND STR_TO_DATE(ENDDATE, '%d,%m,%Y')
    AND idWorkers = IDWORKER;
END //
DELIMITER ;

CALL mydb.getAllDRForWorkBetweenDate ('2,1,2015', '18,1,2022',
13);

```

**Wycinek 14** Procedura pobierająca wszystkie karty pracy pracownika w przedziale czasowym.

	idDailyRaport	Date	idWorkers	idProjects
►	17	2015-01-02	13	1
	45	2015-01-04	13	1
	168	2015-01-14	13	1
	185	2015-01-17	13	2
	222	2015-01-22	13	2
	308	2015-02-02	13	3
	363	2015-02-08	13	4
	427	2015-02-15	13	5
	452	2015-02-17	13	5

**Rysunek 13** Rysunek przedstawia fragment rezultatu dla powyższej procedury.

```

DROP PROCEDURE IF EXISTS mydb.updateProjFinanceFromId ;
DELIMITER //
CREATE PROCEDURE updateProjFinanceFromId
(
    IN BudgetN FLOAT,
    IN ExpcetedProfitN FLOAT,
    IN BonusManagerN FLOAT,
    IN PenaltyPerDayN FLOAT,
    IN CostsN FLOAT,
    IN AdditionalCostsN FLOAT,
    IN idProjFin INT
)
BEGIN
    UPDATE ProjFinance SET Budget = BudgetN, ExpectedProfit =
    ExpcetedProfitN, BonusManager = BonusManagerN, PenaltyPerDay =
    PenaltyPerDayN, Costs = CostsN, AdditionalCosts =
    AdditionalCostsN WHERE idProjFinance = idProjFin;
END //
DELIMITER ;
SELECT * FROM ProjFinance where idProjFinance = 85;
CALL mydb.updateProjFinanceFromId
(187554.2,454332.3,5642.3,1.2,56254, 1643.4, 85);
SELECT * FROM ProjFinance where idProjFinance = 85;

```

**Wycinek 15 Procedura aktualizująca dane rekordu w tabeli ProjFinance.**

	idProjFinance	Budget	ExpectedProfit	BonusManager	PenaltyPerDay	Costs	AdditionalCosts
►	85	187554	454332	5642.3	1.2	56254	1643.4

**Rysunek 14 Rysunek przedstawia rezultat dla powyższej procedury.**

```

DROP PROCEDURE IF EXISTS mydb.addConcractToIdWorker ;
DELIMITER //
CREATE PROCEDURE mydb.addConcractToIdWorker
(
    IN DateStartN VARCHAR(10),
    IN DateENDN VARCHAR(10),
    IN PositionN VARCHAR(45),
    IN RateDayN FLOAT,
    IN idWorkersN INT
)
BEGIN
    INSERT INTO mydb.Contracts(DateStart, DateEND, Position,
RateDay, idWorkers) VALUES (STR_TO_DATE(DateStartN,'%d,%m,%Y')
, STR_TO_DATE(DateENDN,'%d,%m,%Y') , PositionN, RateDayN,
idWorkersN);
END //
DELIMITER ;

SELECT * FROM mydb.Contracts WHERE idWorkers = 90 ORDER BY
DateStart DESC;

CALL mydb.addConcractToIdWorker ('2,10,2023', '25,1,2024',
'Database specialist', 450.23, 90);

SELECT * FROM mydb.Contracts WHERE idWorkers = 90 ORDER BY
DateStart DESC;

```

**Wycinek 16 Procedura dodająca nową umowę do pracownika.**

	idContracts	DateStart	DateEND	Position	RateDay	idWorkers
►	1911	2023-10-02	2024-01-25	Database specialist	450.23	90

**Rysunek 15 Rysunek przedstawia rezultat dla powyższej procedury.**

```

DROP PROCEDURE IF EXISTS
mydb.deleteContractidWorkersDateStartEnd ;
DELIMITER //
CREATE PROCEDURE mydb.deleteContractidWorkersDateStartEnd
(
    IN DateStartN VARCHAR(10),
    IN DateENDN VARCHAR(10),
    IN idWorkersN INT
)
BEGIN
    DELETE FROM mydb.Contracts
WHERE idWorkers = idWorkersN
AND DateStart = STR_TO_DATE(DateStartN, '%d,%m,%Y')
AND DateEND = STR_TO_DATE(DateENDN, '%d,%m,%Y');
END //
DELIMITER ;
SELECT * FROM mydb.Contracts WHERE idWorkers = 90
ORDER BY DateStart DESC;
CALL mydb.deleteContractidWorkersDateStartEnd ('2,10,2023',
'25,1,2024', 90);
SELECT * FROM mydb.Contracts WHERE idWorkers = 90
ORDER BY DateStart DESC;

```

**Wycinek 17 Procedura usuwająca umowę pracownika.**

```
CALL mydb.deleteContractidWorkersDateStartEnd ('2,10,2023', '25,1,2024', 90)
```

```
1 row(s) affected,
```

**Rysunek 16 Rysunek przedstawia rezultat dla powyższej procedury.**

```

DROP PROCEDURE IF EXISTS mydb.sumWorkersRentForProject ;
DELIMITER //
CREATE PROCEDURE sumWorkersRentForProject
(IN InputIdProject INT)
BEGIN
    DECLARE spDate VARCHAR(10);
    DECLARE epDate VARCHAR(10);

    SET @spDate = (SELECT DateStart FROM Projects WHERE
idProjects = InputIdProject);
    SET @epDate = (SELECT DateEnd FROM Projects WHERE
idProjects = InputIdProject);

    SELECT ROUND(SUM(dd.RateDay), 2) FROM ( SELECT DISTINCT
dp.Date, dp.idWorkers, c.RateDay FROM DailyRaport dp
JOIN mydb.Contracts c ON c.idWorkers = dp.idWorkers
WHERE Date BETWEEN STR_TO_DATE(@spDate,'%Y-%m-%d')
AND STR_TO_DATE(@epDate,'%Y-%m-%d')ORDER BY dp.Date) AS dd;
END //
DELIMITER ;

CALL mydb.sumWorkersRentForProject (17);

```

**Wycinek 18** Procedura wyświetlająca kwotę jaką zarobili pracownicy podczas projektu.

	ROUND(SUM(dd.RateDay), 2)
▶	36724.01

**Rysunek 17** Rysunek przedstawia rezultat dla powyższej procedury.