A PROJECT REPORT

ON

INSURANCE DATABASE MANAGEMENT SYSTEM

Submitted by:

102106029	Dhruv Bobal	3EC2
102106069	Pearl Bajpai	3EC3
102106086	Vansh Tejwani	3EC3

Submitted to:

Mr. Rakesh Kumar

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

TABLE OF CONTENTS

Contents

AIM:

The aim of our project - Insurance Database Management System, is to create a database schema for an insurance project and populate it with sample data. It also includes several SQL queries to retrieve specific information from the database.

ABSTRACT:

The goal of this project is to create a complete Insurance Database Management System (IDMS). To do this, an efficient database schema designed for insurance processes will be created, and sample data will be added to it. The main goal is to make insurance-related tasks easier by setting up an organised place to store information about policyholders, policies, claims, and other important records. The system also includes a set of SQL queries that can quickly and correctly get specific information from the database. Using current database management techniques and tools, this project aims to improve the speed, accuracy, and usability of data management tasks related to insurance. This will lead to better insurance-related decision-making.

INTRODUCTION-:

The script begins by creating the SQL_project database named SQL_project and then switches to that database.

It creates tables such as ADDRESS, AGENT, CUSTOMER, Agnt_password, Cust_password, POLICY, Payment, Claim, and cust_policy, each representing different entities within the insurance system.

Foreign key constraints are added to maintain referential integrity between tables.

Sample data is inserted into these tables using INSERT INTO statements.

Various SQL queries are provided to retrieve specific information from the populated database, such as agents working in Florida, customers with active policies, customers who have claimed their policies, customers who have made multiple payments, names and passwords of agents, agents working in Texas, the maximum and second-highest salary of agents, customers associated with agents, and the count of people working in Florida.

HARDWARE & SOFTWARE REQUIREMENTS-:

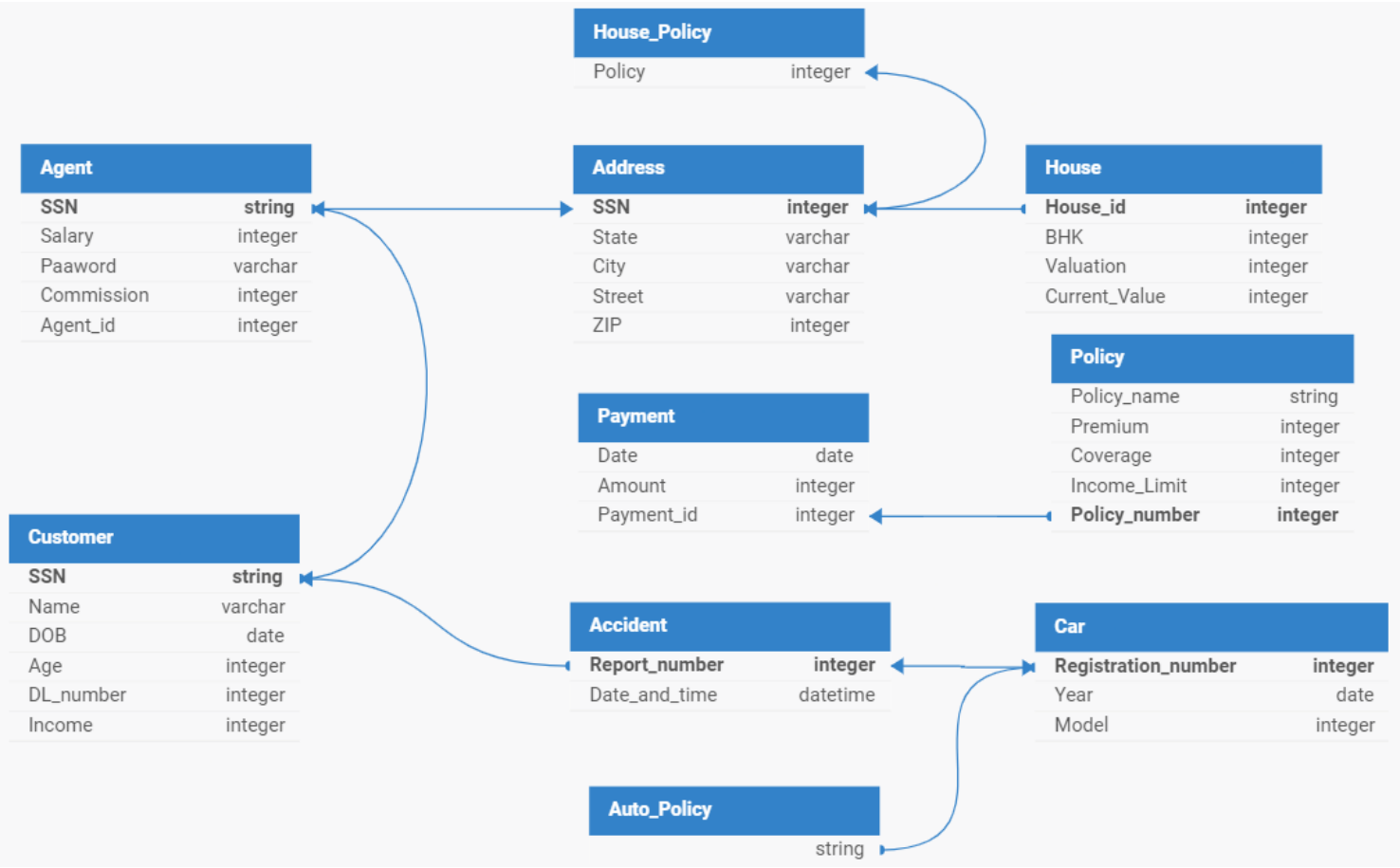
Hardware Requirements-:

Reliable network connectivity to ensure seamless communication between client applications and the database server. A dedicated server or cloud-based infrastructure capable of hosting the database management system.

Software Requirements-:

Any operating system (Windows/IOS/LINUX) with a PL SQL Server or free online PL SQL server like Oracle Live SQL. Integrated Development Environment (IDE) or text editors for writing SQL queries and managing database schema. Version control systems (e.g., Git) for collaborative development and code management.

ER DIAGRAM TO TABLE:



APPROACH:

In this project, several tables are created to facilitate the management of an insurance system.

The ADDRESS Table stores the address details of insurance agents. It includes SSN (foreign key), City, Zip, and State attributes. This table is integrated with the Agent table through the SSN attribute, enabling retrieval of address details for each agent.

The AGENT Table contains information about insurance agents, including attributes like Name, SSN (primary key), Salary, Commission, and Num_of_policy_done. SSN acts as a primary key and is referenced in other tables like Address and Agnt_password.

Customers who purchase insurance policies are managed through the CUSTOMER Table. It includes attributes such as SSN (primary key), Fname, Minit, Lname, DOB, Income, Policy_num, Phone, Agent_id, Purchase_date, and Claim_num. This table is linked with other tables such as Cust_password, Address (through Agent_id), Claim, Cust_policy, and Payment.

The Agnt_password Table and Cust_password Table store passwords for agents and customers respectively. They include attributes like SSN (foreign key) and Password, with SSN acting as a foreign key and relating to the SSN attribute in the Agent and Customer tables.

Details about different insurance policies offered are stored in the POLICY Table, which includes attributes like Policy_id (primary key), Name, Type, Term_price, Term_period, and Coverage. This table is linked with the Cust_policy table through Policy_id.

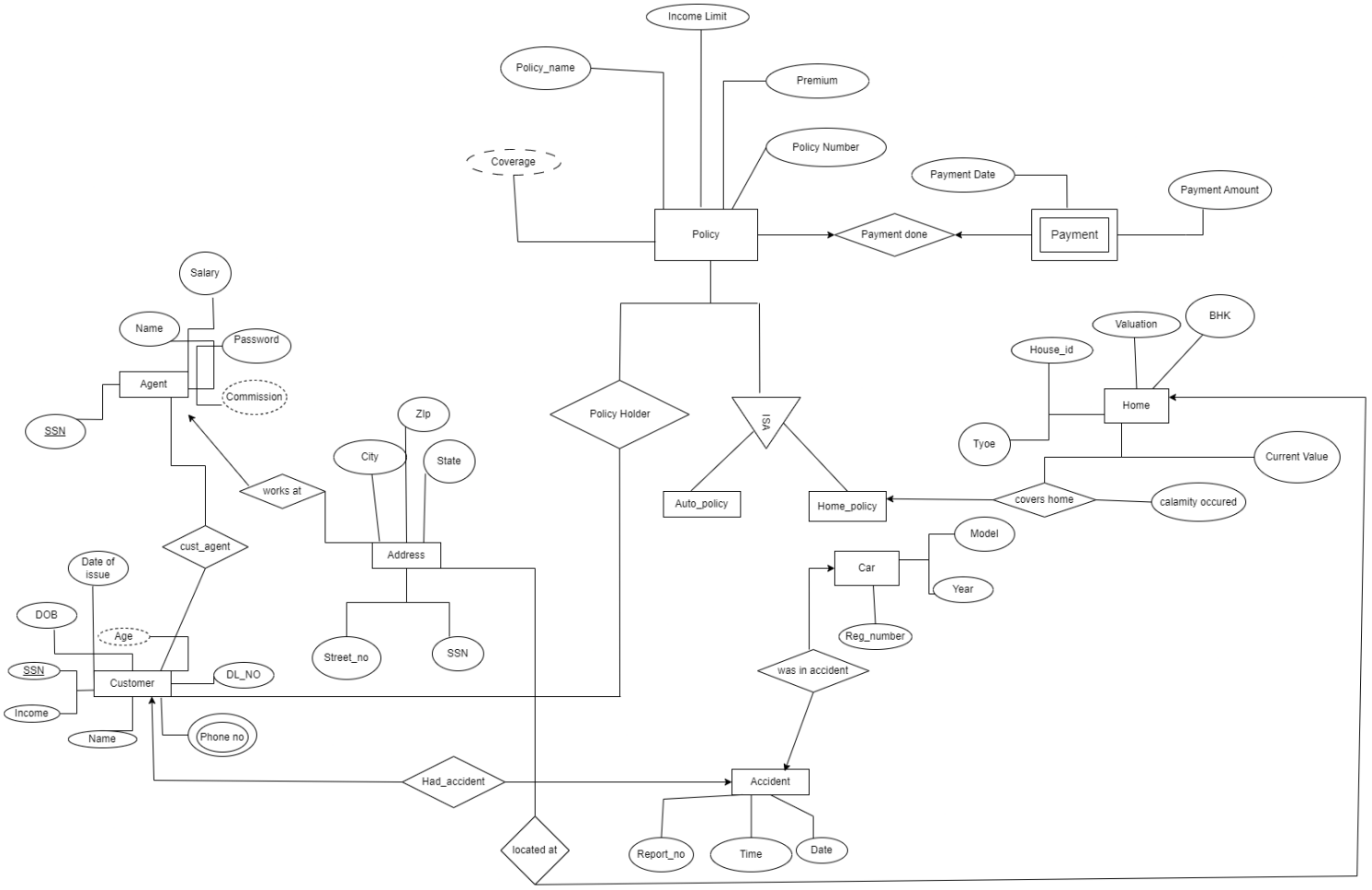
The Payment Table records payments made by customers, with attributes including Payment_no (primary key), SSN (foreign key), Payment_data, Amount, and Policy_num. It is connected with the Customer table through SSN and Policy_num attributes.

The Claim Table stores details about insurance claims made by customers, with attributes such as Claim_id (primary key), Policy_num, Claim_date, Sanc_date, Sanc_amount, Cheaque_no, and Reason. The Policy_num attribute links to the Policy table.

Finally, the cust_policy Table manages the association between customers and their policies. It includes attributes like Policy_num (primary key), Policy_id, Type, Registration_no, Valuation, and Sum_assured. The Policy_num links to the Customer and Policy tables.

These tables are interconnected through primary and foreign key relationships, ensuring data integrity and enabling efficient querying to extract meaningful insights from the database. The integration of these tables forms a comprehensive insurance management system, facilitating operations such as policy management, customer interactions, payment tracking, and claim processing.

ER DIAGRAM:


Income Limit
Policy_name
Premium
Policy Number
Coverage
Payment Date
Payment Amount
Policy
Payment done
Payment
Salary
Name
Password
Agent
Commission
SSN
Zip
City
State
works at
Policy Holder
VSI
Valuation
BHK
House_id
Home
Tyoe
Current Value
Auto_policy
Home_policy
covers home
calamity occured
Model
Car
Year
Reg_number
was in accident
cust_agent
Address
Date of issue
DOB
Age
SSN
Customer
DL_NO
Income
Name
Phone no
Street_no
SSN
Had_accident
Accident
located at
Report_no
Time
Date

CREATION OF TABLES:

```
create table ADDRESS
(
        Ssn CHAR(10) NOT NULL PRIMARY KEY,
        City varchar(7),
        Zip Int,
        State varchar(15)
);


CREATE TABLE AGENT
(
        Name varchar(35),
        Ssn CHAR(10) NOT NULL PRIMARY KEY,
        Salary decimal(12,2),
        Commission decimal(12,2),
        Num_of_policy_done int
);


create table CUSTOMER
(
```

```
        Ssn CHAR(10) NOT NULL PRIMARY KEY,
        Fname VARCHAR(15) NOT NULL ,
        Minit CHAR,
        Lname VARCHAR(15) NOT NULL,
        DOB date,
        Income decimal(12,2),
        Policy_num int unique,
        Phone int,
        Agent_id char(10),
        Purchase_date date,
        Claim_num int
);

create table Agnt_password
(
        Ssn char(10) primary key,
        Password varchar(15) unique
);

create table Cust_password
```

```
(
        Ssn char(10) primary key,
        Password varchar(15) unique
);
create table POLICY
(
        Policy_id int primary key,
        Name varchar(25),
        Type varchar(10),
        Term_price decimal(9,2),
        Term_period int,
        Coverage decimal(12,2)
);
create table Payment
(
        Payment_no int primary key,
        Ssn char(10),
        Payment_data date,
        Amount decimal(8,2),
        Policy_num int
```

```
);
create table Claim
(
        Claim_id int primary key,
        Policy_num int,
        Claim_date date,
        Sanc_date date,
        Sanc_amount decimal(7,2),
        Cheaque_no int,
        Reason varchar(255)
);
create table cust_policy
(
        Policy_num int primary key,
        Policy_id int,
        Type varchar(15),
        Registration_no NUMBER(15),
        Valuation decimal(12,2),
        Sum_assured decimal(12,2)
);
```

Foreign Key Creation

```
/*use information_schema;

select * from key_column_usage where constraint_schema = 'insurance';*/

alter table customer add constraint fk1 foreign key (agent_id) references agent(Ssn);


alter table address add constraint fk2 foreign key(ssn) references agent(Ssn);


alter table cust_policy add constraint fk3 foreign key(Policy_num) references
customer(Policy_num);


alter table Payment add constraint fk4 foreign key(Ssn) references customer(ssn);


alter table customer add constraint fk5 foreign key(claim_num) references claim(claim_id);


alter table cust_policy add constraint fk6 foreign key(policy_id) references policy(policy_id);


alter table Agnt_password add constraint fk7 foreign key(ssn) references agent(ssn);


alter table Cust_password add constraint fk8 foreign key(ssn) references customer(ssn);
```

INSERTION OF DATA:

```
CREATE OR REPLACE PROCEDURE insert_agent(
    agent_name IN VARCHAR2,
    agent_id IN NUMBER,
    sales_volume IN NUMBER,
    commission IN NUMBER,
    experience_years IN NUMBER
)
AS
BEGIN
    INSERT INTO agent VALUES (agent_name, agent_id, sales_volume, commission,
experience_years);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Data inserted successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
BEGIN
```

insert_agent('Obidiah Heathorn', 1000000001, 94200, 17100, 12);

insert_agent('Ricca Artinstall', 1000000002, 119400, 11300, 9);

insert_agent('Bernardine Tayloe',1000000003,157900,34900,9);

insert_agent('Murray Philippou',1000000004,70000,12100,17);

insert_agent('Berrie Chaudrelle',1000000005,158700,33100,14);

insert_agent('Minetta Hodjetts',1000000006,87900,7000,23);

insert_agent('Marylee Kinnach',1000000007,111800,28200,9);

insert_agent('Curtis Clausen-Thue',1000000008,131000,8800,20);

insert_agent('Peterus Derwin',1000000009,105500,14500,20);

insert_agent('Farlay Varfalameev',1000000010,178000,14000,22);

insert_agent('Harmony Ewles',1000000011,81100,34800,7);

insert_agent('Simonette Odby',1000000012,135800,20000,11);

insert_agent('Lucille Skein',1000000013,138500,27300,16);

insert_agent('Katrina Leaburn',1000000014,74700,10400,8);

insert_agent('Othilie Eggleson',1000000015,105500,23900,9);

insert_agent('Zak Fleetham',1000000017,126100,9300,7);

insert_agent('Tine Robert',1000000018,116100,27400,13);

insert_agent('Tatiania Dashwood',1000000019,120700,14600,22);

insert_agent('Kalli Mould',1000000020,114500,22500,14);

insert_agent('Stacia Egalton',1000000021,78900,12800,19);

```
insert_agent('Hollis Persich',1000000022,125400,16100,5);

insert_agent('Bonny Janicijevic',1000000023,106600,15400,6);

insert_agent('Kane Grcic',1000000024,99600,26500,15);

insert_agent('Etan Bernardez',1000000025,139500,31700,11);

insert_agent('Darb Spriggin',1000000026,85700,5400,5);

insert_agent('See Musso',1000000027,90100,23800,10);

insert_agent('Michelina Plank',1000000028,171300,12900,12);

insert_agent('Domenic Wooffitt',1000000029,94900,13000,7);

insert_agent('Standford Scarre',1000000030,106900,22900,14);

insert into agent values('Elyse Wells',1000000016,78700,7200,14);

   -- Call insert_agent for each set of values you want to insert

END;

/
```

select * from agent;

NAME	SSN	SALARY	COMMISSION	NUM_OF_POLICY_DONE
Obidiah Heathorn	1000000001	94200	17100	12
Ricca Artinstall	1000000002	119400	11300	9
Bernardine Tayloe	1000000003	157900	34900	9
Murray Philippou	1000000004	70000	12100	17
Berrie Chaudrelle	1000000005	158700	33100	14
Minetta Hodjetts	1000000006	87900	7000	23
Marylee Kinnach	1000000007	111800	28200	9
Curtis Clausen-Thue	1000000008	131000	8800	20
Peterus Derwin	1000000009	105500	14500	20
Farlay Varfalameev	1000000010	178000	14000	22
Harmony Ewles	1000000011	81100	34800	7
Simonette Odby	1000000012	135800	20000	11

Lucille Skein	1000000013	138500	27300	16
Katrina Leaburn	1000000014	74700	10400	8
Othilie Eggleson	1000000015	105500	23900	9
Zak Fleetham	1000000017	126100	9300	7
Tine Robert	1000000018	116100	27400	13
Tatiania Dashwood	1000000019	120700	14600	22
Kalli Mould	1000000020	114500	22500	14
Stacia Egalton	1000000021	78900	12800	19
Hollis Persich	1000000022	125400	16100	5
Bonny Janicijevic	1000000023	106600	15400	6
Kane Grcic	1000000024	99600	26500	15

Etan Bernardez	1000000025	139500	31700	11
Darb Spriggin	1000000026	85700	5400	5
See Musso	1000000027	90100	23800	10
Michelina Plank	1000000028	171300	12900	12
Domenic Wooffitt	1000000029	94900	13000	7
Standford Scarre	1000000030	106900	22900	14
Elyse Wells	1000000016	78700	7200	14

```
CREATE OR REPLACE PROCEDURE insert_address(
   ssn_in IN CHAR,
   city_in IN VARCHAR2,
   zip_in IN INT,
   state_in IN VARCHAR2
)
AS
BEGIN
   INSERT INTO ADDRESS VALUES (ssn_in, city_in, zip_in, state_in);
   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Data inserted successfully.');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
   ALTER TABLE ADDRESS MODIFY (City VARCHAR2(20));
ALTER TABLE ADDRESS MODIFY  (STATE VARCHAR(20));


BEGIN
```

```
 insert_address(1000000001,'Miami',33283,'Florida');

insert_address(1000000002,'Lafayette',47905,'Indiana');

insert_address(1000000003,'Dallas',75287,'Texas');

insert_address(1000000004,'New York City',10110,'New York');

insert_address(1000000005,'Tampa',33647,'Florida');

insert_address(1000000006,'Atlanta',31119,'Georgia');

insert_address(1000000007,'Boise',83727,'Idaho');

insert_address(1000000008,'Tampa',33633,'Florida');

insert_address(1000000009,'Melbourne',32941,'Florida');

insert_address(1000000010,'Dallas',75387,'Texas');

insert_address(1000000011,'Ridgely',21684,'Maryland');

insert_address(1000000012,'Phoenix',85030,'Arizona');

insert_address(1000000013,'Charlotte',28205,'North Carolina');

insert_address(1000000014,'San Francisco',94164,'California');

insert_address(1000000015,'Lansing',48930,'Michigan');

insert_address(1000000016,'Tampa',33605,'Florida');

insert_address(1000000017,'Jacksonville',32220,'Florida');

insert_address(1000000018,'Richmond',23237,'Virginia');

insert_address(1000000019,'Detroit',48211,'Michigan');

insert_address(1000000020,'San Jose',95123,'California');
```

```
insert_address(1000000021,'Austin',78732,'Texas');

insert_address(1000000022,'Washington',20525,'District of Columbia');

insert_address(1000000023,'San Francisco',94137,'California');

insert_address(1000000024,'Charlottesville',22908,'Virginia');

insert_address(1000000025,'Roanoke',24040,'Virginia');

insert_address(1000000026,'Los Angeles',90040,'California');

insert_address(1000000027,'Saint Paul',55127,'Minnesota');

insert_address(1000000028,'Kansas City',64144,'Missouri');

insert_address(1000000029,'Omaha',68117,'Nebraska');

insert_address(1000000030,'Little Rock',72222,'Arkansas');

END;

/
```

select * from ADDRESS;

SSN	CITY	ZIP	STATE
1000000001	Miami	33283	Florida
1000000002	Lafayette	47905	Indiana
1000000003	Dallas	75287	Texas
1000000004	New York City	10110	New York
1000000005	Tampa	33647	Florida
1000000006	Atlanta	31119	Georgia
1000000007	Boise	83727	Idaho
1000000008	Tampa	33633	Florida
1000000009	Melbourne	32941	Florida
1000000010	Dallas	75387	Texas
1000000011	Ridgely	21684	Maryland
1000000012	Phoenix	85030	Arizona

1000000013	Charlotte	28205	North Carolina
1000000014	San Francisco	94164	California
1000000015	Lansing	48930	Michigan
1000000016	Tampa	33605	Florida
1000000017	Jacksonville	32220	Florida
1000000018	Richmond	23237	Virginia
1000000019	Detroit	48211	Michigan
1000000020	San Jose	95123	California
1000000021	Austin	78732	Texas
1000000022	Washington	20525	District of Columbia
1000000023	San Francisco	94137	California
1000000024	Charlottesville	22908	Virginia
1000000025	Roanoke	24040	Virginia
1000000026	Los Angeles	90040	California
1000000027	Saint Paul	55127	Minnesota
1000000028	Kansas City	64144	Missouri
1000000029	Omaha	68117	Nebraska
1000000030	Little Rock	72222	Arkansas

```
insert into agnt_password values(1000000001,'YoJqDBn6L');
insert into agnt_password values(1000000002,'Y2k7u2LfcdFu');
insert into agnt_password values(1000000003,'cyr6XR8qFB');
insert into agnt_password values(1000000004,'N4uoBSL0UQ');
insert into agnt_password values(1000000005,'YW494YQ');
insert into agnt_password values(1000000006,'AW05O1xAc4F3');
insert into agnt_password values(1000000007,'RBpAUoCFWDzh');
insert into agnt_password values(1000000008,'kqcj3ZCacu');
insert into agnt_password values(1000000009,'veu4cx4EotE');
insert into agnt_password values(1000000010,'LI3CmPoWXYN3');
insert into agnt_password values(1000000011,'CWhR7FCr');
insert into agnt_password values(1000000012,'LpHxnYnY');
insert into agnt_password values(1000000013,'UEA8y24aPvV');
insert into agnt_password values(1000000014,'VNyPTwfNvSs');
insert into agnt_password values(1000000015,'idLenTR342');
insert into agnt_password values(1000000016,'evO6x1doy');
insert into agnt_password values(1000000017,'nYNv0tL4gAZ');
insert into agnt_password values(1000000018,'xUdRifCNlT');
insert into agnt_password values(1000000019,'kJ45l7ztNr');
insert into agnt_password values(1000000020,'XgwxAT');
```

```
insert into agnt_password values(1000000021,'GPPUDfhno');

insert into agnt_password values(1000000022,'Z7BIDCX');

insert into agnt_password values(1000000023,'HCun2at88HeF');

insert into agnt_password values(1000000024,'d6ddJwMoSX');

insert into agnt_password values(1000000025,'d6ddJklpd');

insert into agnt_password values(1000000026,'sSkH3CgQ');

insert into agnt_password values(1000000027,'7Na6GF');

insert into agnt_password values(1000000028,'AJTT2b');

insert into agnt_password values(1000000029,'y7kf2NxXqiB');

insert into agnt_password values(1000000030,'mNbo0Q1F');
```

select * from agnt_password;

SSN	PASSWORD
1000000001	YoJqDBn6L
1000000002	Y2k7u2LfcdFu
1000000003	cyr6XR8qFB
1000000004	N4uoBSL0UQ
1000000005	YW494YQ
1000000006	AW05O1xAc4F3
1000000007	RBpAUoCFWDzh
1000000008	kqcj3ZCacu
1000000009	veu4cx4EotE
1000000010	LI3CmPoWXYN3
1000000011	CWhR7FCr
1000000012	LpHxnYnY
1000000013	UEA8y24aPvV
1000000014	VNyPTwfNvSs
1000000015	idLenTR342
1000000016	evO6x1doy
1000000017	nYNv0tL4gAZ
1000000018	xUdRifCNlT
1000000019	kJ45l7ztNr
1000000020	XgwxAT
1000000021	GPPUDfhno
1000000022	Z7BIDCX
1000000023	HCun2at88HeF
1000000024	d6ddJwMoSX
1000000025	d6ddJklpd
1000000026	sSkH3CgQ
1000000027	7Na6GF
1000000028	AJTT2b
1000000029	y7kf2NxXqiB
1000000030	mNbo0Q1F

```
-- Insert into claim table with adjusted date formats

DECLARE

BEGIN

   INSERT INTO claim VALUES (1, 1, TO_DATE('2017-09-01', 'YYYY-MM-DD'),
TO_DATE('2017-09-16', 'YYYY-MM-DD'), 50000, 1234, 'Car accident');

   INSERT INTO claim VALUES (2, 3, TO_DATE('2017-07-01', 'YYYY-MM-DD'),
TO_DATE('2017-07-16', 'YYYY-MM-DD'), 50000, 1235, 'Car accident');

   INSERT INTO claim VALUES (3, 2, TO_DATE('2017-08-01', 'YYYY-MM-DD'),
TO_DATE('2017-08-16', 'YYYY-MM-DD'), 50000, 1239, 'Car accident');

   INSERT INTO claim VALUES (4, 4, TO_DATE('2017-01-01', 'YYYY-MM-DD'),
TO_DATE('2017-01-16', 'YYYY-MM-DD'), 50000, 1236, 'Car accident');

   COMMIT;

END;
```

select * from claim;

CLAIM_ID	POLICY_NUM	CLAIM_DATE	SANC_DATE	SANC_AMOUNT	CHEAQUE_NO	REASON
1	1	01-SEP-17	16-SEP-17	50000	1234	Car accident
2	3	01-JUL-17	16-JUL-17	50000	1235	Car accident
3	2	01-AUG-17	16-AUG-17	50000	1239	Car accident
4	4	01-JAN-17	16-JAN-17	50000	1236	Car accident

```
-- Convert data type of the Phone column

ALTER TABLE CUSTOMER MODIFY (Phone NUMBER(20));

INSERT INTO customer VALUES(1000000031, 'Shaine', 'F', 'Plastow', TO_DATE('1993-02-28', 'YYYY-MM-DD'), 156700, 11, 2191709420, 1000000011, TO_DATE('2018-08-17', 'YYYY-MM-DD'), 1);

INSERT INTO customer VALUES(1000000032, 'Phillida', 'Q', 'Farland', TO_DATE('1994-07-10', 'YYYY-MM-DD'), 119400, 5, 2286461611, 1000000018, TO_DATE('2019-01-19', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000033, 'Maximilian', 'H', 'Deniskevich', TO_DATE('1995-11-02', 'YYYY-MM-DD'), 148500, 6, 6786580941, 1000000014, TO_DATE('2018-11-01', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000034, 'Rosalinda', 'R', 'Bloore', TO_DATE('2001-12-01', 'YYYY-MM-DD'), 137100, 12, 1675878478, 1000000021, TO_DATE('1999-01-13', 'YYYY-MM-DD'), 2);

INSERT INTO customer VALUES(1000000035, 'Gloriana', 'R', 'Gebhardt', TO_DATE('1973-03-30', 'YYYY-MM-DD'), 139200, 14, 9555044231, 1000000020, TO_DATE('2018-10-03', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000036, 'Pierrette', 'P', 'Kilmary', TO_DATE('1980-07-05', 'YYYY-MM-DD'), 137700, 18, 7295006186, 1000000016, TO_DATE('2018-06-05', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000037, 'Melisande', 'U', 'Lyptratt', TO_DATE('1995-04-14', 'YYYY-MM-DD'), 157700, 13, 7124554531, 1000000011, TO_DATE('2018-10-09', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000038, 'Marty', 'C', 'Dalgliesh', TO_DATE('1973-03-22', 'YYYY-MM-DD'), 141100, 21, 8528177396, 1000000017, TO_DATE('2018-03-26', 'YYYY-MM-DD'), null);
```

INSERT INTO customer VALUES(1000000039, 'Karin', 'M', 'Rothermel', TO_DATE('1976-02-22', 'YYYY-MM-DD'), 93300, 15, 3589575033, 1000000029, TO_DATE('2018-08-09', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000040, 'Emmott', 'B', 'Dominicacci', TO_DATE('1977-08-12', 'YYYY-MM-DD'), 146500, 22, 4946543104, 1000000010, TO_DATE('2018-09-01', 'YYYY-MM-DD'), 3);

INSERT INTO customer VALUES(1000000041, 'Jarvis', 'Q', 'Muehler', TO_DATE('1974-09-20', 'YYYY-MM-DD'), 109200, 27, 2033931454, 1000000006, TO_DATE('2018-08-10', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000042, 'Pamela', 'V', 'Castellani', TO_DATE('1985-02-22', 'YYYY-MM-DD'), 140900, 10, 3894717434, 1000000009, TO_DATE('2018-08-22', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000043, 'Orson', 'I', 'Drewe', TO_DATE('1986-01-11', 'YYYY-MM-DD'), 86300, 23, 2096310563, 1000000019, TO_DATE('2018-08-18', 'YYYY-MM-DD'), 4);

INSERT INTO customer VALUES(1000000044, 'Marabel', 'A', 'MacRirie', TO_DATE('1988-07-05', 'YYYY-MM-DD'), 140000, 4, 6066758218, 1000000011, TO_DATE('2018-11-23', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000045, 'Bobine', 'H', 'Purver', TO_DATE('1989-11-25', 'YYYY-MM-DD'), 132100, 1, 8391673429, 1000000015, TO_DATE('2018-12-31', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000046, 'Edouard', 'Z', 'Thompson', TO_DATE('1986-11-22', 'YYYY-MM-DD'), 134100, 16, 1076465899, 1000000025, TO_DATE('2018-11-27', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000047, 'Feliza', 'A', 'Kaplan', TO_DATE('1986-12-21', 'YYYY-MM-DD'), 150400, 2, 7817716875, 1000000003, TO_DATE('2019-01-16', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000048, 'Regina', 'I', 'Alleyne', TO_DATE('1994-12-31', 'YYYY-MM-DD'), 115000, 3, 3162932738, 1000000003, TO_DATE('2018-08-31', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000049, 'Winslow', 'K', 'Argente', TO_DATE('1989-03-19', 'YYYY-MM-DD'), 158200, 8, 3278672633, 1000000014, TO_DATE('2018-06-29', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000050, 'Moises', 'Q', 'McGahern', TO_DATE('1998-03-27', 'YYYY-MM-DD'), 114600, 17, 9087668597, 1000000004, TO_DATE('2018-04-05', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000051, 'Leah', 'S', 'Laterza', TO_DATE('1998-08-22', 'YYYY-MM-DD'), 119600, 19, 8431272892, 1000000026, TO_DATE('2018-08-20', 'YYYY-MM-DD'), 2);

INSERT INTO customer VALUES(1000000052, 'Alanna', 'A', 'Slater', TO_DATE('1990-04-30', 'YYYY-MM-DD'), 112100, 20, 1575107631, 1000000029, TO_DATE('2018-10-16', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000053, 'Judah', 'X', 'Bruffell', TO_DATE('1996-05-24', 'YYYY-MM-DD'), 105900, 24, 1298822583, 1000000012, TO_DATE('2018-12-05', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000054, 'Rancell', 'H', 'Rabb', TO_DATE('1996-02-15', 'YYYY-MM-DD'), 159400, 25, 8398937234, 1000000021, TO_DATE('2018-12-20', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000055, 'Isac', 'S', 'Yacob', TO_DATE('1978-08-07', 'YYYY-MM-DD'), 118300, 7, 1095009928, 1000000019, TO_DATE('2018-03-22', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000056, 'Addi', 'Y', 'McInnerny', TO_DATE('1978-09-25', 'YYYY-MM-DD'), 156700, 26, 8664395157, 1000000002, TO_DATE('2018-10-21', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000057, 'Amil', 'V', 'Solly', TO_DATE('1996-01-15', 'YYYY-MM-DD'), 119900, 28, 9912292272, 1000000011, TO_DATE('2018-03-28', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000058, 'Orton', 'M', 'Dimitrov', TO_DATE('1992-12-15', 'YYYY-MM-DD'), 116700, 29, 6018245559, 1000000025, TO_DATE('2018-06-04', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000059, 'Ugo', 'Q', 'Toffanini', TO_DATE('2000-04-05', 'YYYY-MM-DD'), 104100, 30, 4197448545, 1000000009, TO_DATE('2019-01-19', 'YYYY-MM-DD'), null);

INSERT INTO customer VALUES(1000000060, 'Oswell', 'T', 'Sango', TO_DATE('1972-05-07', 'YYYY-MM-DD'), 82000, 31, 6162363629, 1000000023, TO_DATE('2018-11-23', 'YYYY-MM-DD'), null);

select * from customer;

SSN	FNAME	MINIT	LNAME	DOB	INCOME	POLICY_NUM	PHONE	AGENT_ID	PURCHASE_DATE	CLAIM_NUM
1000000031	Shaine	F	Plastow	28-FEB-93	156700	11	2191709420	1000000011	17-AUG-18	1
1000000032	Phillida	Q	Farland	10-JUL-94	119400	5	2286461611	1000000018	19-JAN-19	-
1000000033	Maximilian	H	Deniskevich	02-NOV-95	148500	6	6786580941	1000000014	01-NOV-18	-
1000000034	Rosalinda	R	Bloore	01-DEC-01	137100	12	1675878478	1000000021	13-JAN-99	2
1000000035	Gloriana	R	Gebhardt	30-MAR-73	139200	14	9555044231	1000000020	03-OCT-18	-
1000000036	Pierrette	P	Kilmary	05-JUL-80	137700	18	7295006186	1000000016	05-JUN-18	-
1000000037	Melisande	U	Lyptratt	14-APR-95	157700	13	7124554531	1000000011	09-OCT-18	-
1000000038	Marty	C	Dalgliesh	22-MAR-73	141100	21	8528177396	1000000017	26-MAR-18	-
1000000039	Karin	M	Rothermel	22-FEB-76	93300	15	3589575033	1000000029	09-AUG-18	-
1000000040	Emmott	B	Dominicacci	12-AUG-77	146500	22	4946543104	1000000010	01-SEP-18	3
1000000041	Jarvis	Q	Muehler	20-SEP-74	109200	27	2033931454	1000000006	10-AUG-18	-
1000000042	Pamela	V	Castellani	22-FEB-85	140900	10	3894717434	1000000009	22-AUG-18	-
1000000043	Orson	I	Drewe	11-JAN-86	86300	23	2096310563	1000000019	18-AUG-18	4
1000000044	Marabel	A	MacRirie	05-JUL-88	140000	4	6066758218	1000000011	23-NOV-18	-
1000000045	Bobine	H	Purver	25-NOV-89	132100	1	8391673429	1000000015	31-DEC-18	-
1000000046	Edouard	Z	Thompson	22-NOV-86	134100	16	1076465899	1000000025	27-NOV-18	-
1000000047	Feliza	A	Kaplan	21-DEC-86	150400	2	7817716875	1000000003	16-JAN-19	-
1000000048	Regina	I	Alleyne	31-DEC-94	115000	3	3162932738	1000000003	31-AUG-18	-
1000000049	Winslow	K	Argente	19-MAR-89	158200	8	3278672633	1000000014	29-JUN-18	-
1000000050	Moises	Q	McGahern	27-MAR-98	114600	17	9087668597	1000000004	05-APR-18	-
1000000051	Leah	S	Laterza	22-AUG-98	119600	19	8431272892	1000000026	20-AUG-18	2
1000000052	Alanna	A	Slater	30-APR-90	112100	20	1575107631	1000000029	16-OCT-18	-
1000000053	Judah	X	Bruffell	24-MAY-96	105900	24	1298822583	1000000012	05-DEC-18	-
1000000054	Rancell	H	Rabb	15-FEB-96	159400	25	8398937234	1000000021	20-DEC-18	-
1000000055	Isac	S	Yacob	07-AUG-78	118300	7	1095009928	1000000019	22-MAR-18	-
1000000056	Addi	Y	McInnerny	25-SEP-78	156700	26	8664395157	1000000002	21-OCT-18	-
1000000057	Amil	V	Solly	15-JAN-96	119900	28	9912292272	1000000011	28-MAR-18	-
1000000058	Orton	M	Dimitrov	15-DEC-92	116700	29	6018245559	1000000025	04-JUN-18	-
1000000059	Ugo	Q	Toffanini	05-APR-00	104100	30	4197448545	1000000009	19-JAN-19	-
1000000060	Oswell	T	Sango	07-MAY-72	82000	31	6162363629	1000000023	23-NOV-18	-

ALTER TABLE policy MODIFY (Term_price decimal(12,2));

insert into policy values(1,'home_pol','home',1000000.00,1,100000000.00);

insert into policy values(2,'health_ins','personal',100000.00,3,1000000.00);

insert into policy values(3,'car_pol','car',10000000,1,10000000.00);

insert into policy values(4,'crop_ins','farmer',500000.00,1,10000000.00);

insert into policy values(5,'trvl_ins','travel',500000.00,1,10000000.00);

insert into policy values(6,'life_ins','personal',1000000.00,60,10000000.00);

select * from policy;

POLICY_ID	NAME	TYPE	TERM_PRICE	TERM_PERIOD	COVERAGE
1	home_pol	home	1000000	1	100000000
2	health_ins	personal	100000	3	1000000
3	car_pol	car	10000000	1	10000000
4	crop_ins	farmer	500000	1	10000000
5	trvl_ins	travel	500000	1	10000000
6	life_ins	personal	1000000	60	10000000

```
insert into cust_password values(1000000031,'YoJqDBn6L');
insert into cust_password values(1000000032,'Y2k7u2LfcdFu');
insert into cust_password values(1000000033,'cyr6XR8qFB');
insert into cust_password values(1000000034,'N4uoBSL0UQ');
insert into cust_password values(1000000035,'YW494YQ');
insert into cust_password values(1000000036,'AW05O1xAc4F3');
insert into cust_password values(1000000037,'RBpAUoCFWDzh');
insert into cust_password values(1000000038,'kqcj3ZCacu');
insert into cust_password values(1000000039,'veu4cx4EotE');
insert into cust_password values(1000000040,'LI3CmPoWXYN3');
insert into cust_password values(1000000041,'CWhR7FCr');
insert into cust_password values(1000000042,'teTX1FyaEb');
insert into cust_password values(1000000043,'H9CJW6kYNa');
insert into cust_password values(1000000044,'4a2mIQTy8SjS');
insert into cust_password values(1000000045,'JhIPWE');
insert into cust_password values(1000000046,'NVosMsqFz2g');
insert into cust_password values(1000000047,'I7COibx8Y');
insert into cust_password values(1000000048,'lMc1MO');
insert into cust_password values(1000000049,'3ehpbR6mkup');
insert into cust_password values(1000000050,'R7g2wD4yRa');
```

```
insert into cust_password values(1000000051,'KJYe9ZQ7');

insert into cust_password values(1000000052,'MfSsUf2p9');

insert into cust_password values(1000000053,'cJQb0aiI4');

insert into cust_password values(1000000054,'pdKQE14');

insert into cust_password values(1000000055,'qBXb7xNk');

insert into cust_password values(1000000056,'roCxj6TYa');

insert into cust_password values(1000000057,'U1ma4Q8u');

insert into cust_password values(1000000058,'Nm0GpdyhI3N');

insert into cust_password values(1000000059,'gyL49k');

insert into cust_password values(1000000060,'cu38wz');
```

select * from cust_password;

SSN	PASSWORD
1000000031	YoJqDBn6L
1000000032	Y2k7u2LfcdFu
1000000033	cyr6XR8qFB
1000000034	N4uoBSL0UQ
1000000035	YW494YQ
1000000036	AW05O1xAc4F3
1000000037	RBpAUoCFWDzh
1000000038	kqcj3ZCacu
1000000039	veu4cx4EotE
1000000040	LI3CmPoWXYN3
1000000041	CWhR7FCr
1000000042	teTX1FyaEb
1000000043	H9CJW6kYNa
1000000044	4a2mIQTy8SjS
1000000045	JhIPWE
1000000046	NVosMsqFz2g
1000000047	I7COibx8Y
1000000048	lMc1MO
1000000049	3ehpbR6mkup
1000000050	R7g2wD4yRa
1000000051	KJYe9ZQ7
1000000052	MfSsUf2p9
1000000053	cJQb0aiI4
1000000054	pdKQE14
1000000055	qBXb7xNk
1000000056	roCxj6TYa
1000000057	U1ma4Q8u
1000000058	Nm0GpdyhI3N
1000000059	gyL49k
1000000060	cu38wz

```
ALTER TABLE cust_policy MODIFY Registration_no NUMBER(19);

insert into cust_policy values (1, 1, 'home', 752448054743, 8175000,0);

insert into cust_policy values (2, 2, 'personal', 425245882758, 2500000,0);

insert into cust_policy values (3, 6, 'personal', 904797817681, 500000,0);

insert into cust_policy values (4, 4, 'farmer', 117055384636, 500000,0);

insert into cust_policy values (5, 4, 'farmer', 647568011156, 347000,0);

insert into cust_policy values (6, 4, 'farmer', 901427160928, 120000,0);

insert into cust_policy values (7, 5, 'travel', 465796453800, 475000,0);

insert into cust_policy values (8, 3, 'car', 355224621405, 347000,0);

insert into cust_policy values (31, 4, 'farmer', 353188042827, 85000,0);

insert into cust_policy values (10, 4, 'farmer', 225007951682, 188000,0);

insert into cust_policy values (11, 4, 'farmer', 398537310531, 142000,0);

insert into cust_policy values (12, 4, 'farmer', 306269929682, 44461,0);

insert into cust_policy values (13, 1, 'home', 554122575857, 5721705,0);

insert into cust_policy values (14, 1, 'home', 355127906516, 4401303,0);

insert into cust_policy values (15, 4, 'farmer', 277226711134, 74049,0);

insert into cust_policy values (16, 5, 'travel', 222961787364, 1566183,0);

insert into cust_policy values (17, 4, 'farmer', 658783796475, 92459,0);

insert into cust_policy values (18, 3, 'car', 205082957473, 833228,0);

insert into cust_policy values (19, 4, 'farmer', 850347862971, 62903,0);
```

```
insert into cust_policy values (20, 6, 'personal', 645489320271, 506469,0);

insert into cust_policy values (21, 5, 'travel', 759429136143, 2018749,0);

insert into cust_policy values (22, 2, 'personal', 505396029008, 700000,0);

insert into cust_policy values (23, 5, 'travel', 824766789003, 1500000,0);

insert into cust_policy values (24, 1, 'home', 775256920591, 7500000,0);

insert into cust_policy values (25, 1, 'home', 765116884416, 8475000,0);

insert into cust_policy values (26, 5, 'travel', 176294742712, 1757000,0);

insert into cust_policy values (27, 1, 'home', 354460846922, 8400000,0);

insert into cust_policy values (28, 2, 'personal', 305687141564, 200000,0);

insert into cust_policy values (29, 6, 'personal', 271304288894, 500000,0);

insert into cust_policy values (30, 3, 'car', 697859577691, 1000000,0);
```

select * from cust_policy;

POLICY_NUM	POLICY_ID	TYPE	REGISTRATION_NO	VALUATION	SUM_ASSURED
1	1	home	752448054743	8175000	0
2	2	personal	425245882758	2500000	0
3	6	personal	904797817681	500000	0
4	4	farmer	117055384636	500000	0
5	4	farmer	647568011156	347000	0
6	4	farmer	901427160928	120000	0
7	5	travel	465796453800	475000	0
8	3	car	355224621405	347000	0
31	4	farmer	353188042827	85000	0
10	4	farmer	225007951682	188000	0
11	4	farmer	398537310531	142000	0
12	4	farmer	306269929682	44461	0

13	1	home	554122575857	5721705	0
14	1	home	355127906516	4401303	0
15	4	farmer	277226711134	74049	0
16	5	travel	222961787364	1566183	0
17	4	farmer	658783796475	92459	0
18	3	car	205082957473	833228	0
19	4	farmer	850347862971	62903	0
20	6	personal	645489320271	506469	0
21	5	travel	759429136143	2018749	0
22	2	personal	505396029008	700000	0
23	5	travel	824766789003	1500000	0
24	1	home	775256920591	7500000	0
25	1	home	765116884416	8475000	0
26	5	travel	176294742712	1757000	0
27	1	home	354460846922	8400000	0
28	2	personal	305687141564	200000	0
29	6	personal	271304288894	500000	0
30	3	car	697859577691	1000000	0

```
insert into payment values(1,1000000031,to_date('2016-09-08', 'YYYY-MM-DD'),12589,1);
insert into payment values(2,1000000032,to_date('2016-08-25', 'YYYY-MM-DD'),8454,2);
insert into payment values(3,1000000033,to_date('2016-12-18', 'YYYY-MM-DD'),8998,3);
insert into payment values(4,1000000034,to_date('2016-03-01', 'YYYY-MM-DD'),6707,4);
insert into payment values(5,1000000035,to_date('2016-12-22', 'YYYY-MM-DD'),14123,5);
insert into payment values(6,1000000036,to_date('2016-03-04', 'YYYY-MM-DD'),7050,6);
insert into payment values(7,1000000037,to_date('2016-09-20', 'YYYY-MM-DD'),12509,7);
insert into payment values(8,1000000038,to_date('2016-11-08', 'YYYY-MM-DD'),14778,8);
insert into payment values(9,1000000039,to_date('2016-05-19', 'YYYY-MM-DD'),5246,9);
insert into payment values(10,1000000040,to_date('2016-06-23', 'YYYY-MM-DD'),5253,10);
insert into payment values(11,1000000041,to_date('2016-12-06', 'YYYY-MM-DD'),13728,11);
insert into payment values(12,1000000042,to_date('2016-07-28', 'YYYY-MM-DD'),13118,12);
insert into payment values(13,1000000043,to_date('2016-07-05', 'YYYY-MM-DD'),13241,13);
insert into payment values(14,1000000044,to_date('2016-11-22', 'YYYY-MM-DD'),7614,14);
insert into payment values(15,1000000045,to_date('2016-01-31', 'YYYY-MM-DD'),9923,15);
insert into payment values(16,1000000046,to_date('2016-08-07', 'YYYY-MM-DD'),11678,16);
insert into payment values(17,1000000047,to_date('2016-01-14', 'YYYY-MM-DD'),13419,17);
insert into payment values(18,1000000048,to_date('2016-05-20', 'YYYY-MM-DD'),6482,18);
insert into payment values(19,1000000049,to_date('2016-03-01', 'YYYY-MM-DD'),11443,19);
insert into payment values(20,1000000050,to_date('2016-04-08', 'YYYY-MM-DD'),11972,20);
```

```
insert into payment values(21,1000000051,to_date('2016-09-10', 'YYYY-MM-DD'),5413,21);

insert into payment values(22,1000000052,to_date('2016-11-05', 'YYYY-MM-DD'),6453,22);

insert into payment values(23,1000000053,to_date('2016-02-29', 'YYYY-MM-DD'),10491,23);

insert into payment values(24,1000000054,to_date('2016-08-02', 'YYYY-MM-DD'),11189,24);

insert into payment values(25,1000000055,to_date('2016-04-10', 'YYYY-MM-DD'),11630,25);

insert into payment values(26,1000000056,to_date('2016-11-18', 'YYYY-MM-DD'),11558,26);

insert into payment values(27,1000000057,to_date('2016-11-21', 'YYYY-MM-DD'),11166,27);

insert into payment values(28,1000000058,to_date('2016-11-16', 'YYYY-MM-DD'),12941,28);

insert into payment values(29,1000000059,to_date('2016-11-11', 'YYYY-MM-DD'),14241,29);

insert into payment values(30,1000000060,to_date('2016-04-06', 'YYYY-MM-DD'),9391,30);
```

select * from payment;

PAYMENT_NO	SSN	PAYMENT_DATA	AMOUNT	POLICY_NUM
1	1000000031	08-SEP-16	12589	1
2	1000000032	25-AUG-16	8454	2
3	1000000033	18-DEC-16	8998	3
4	1000000034	01-MAR-16	6707	4
5	1000000035	22-DEC-16	14123	5
6	1000000036	04-MAR-16	7050	6
7	1000000037	20-SEP-16	12509	7
8	1000000038	08-NOV-16	14778	8
9	1000000039	19-MAY-16	5246	9
10	1000000040	23-JUN-16	5253	10
11	1000000041	06-DEC-16	13728	11
12	1000000042	28-JUL-16	13118	12

13	1000000043	05-JUL-16	13241	13
14	1000000044	22-NOV-16	7614	14
15	1000000045	31-JAN-16	9923	15
16	1000000046	07-AUG-16	11678	16
17	1000000047	14-JAN-16	13419	17
18	1000000048	20-MAY-16	6482	18
19	1000000049	01-MAR-16	11443	19
20	1000000050	08-APR-16	11972	20
21	1000000051	10-SEP-16	5413	21
22	1000000052	05-NOV-16	6453	22
23	1000000053	29-FEB-16	10491	23
24	1000000054	02-AUG-16	11189	24
25	1000000055	10-APR-16	11630	25
26	1000000056	18-NOV-16	11558	26
27	1000000057	21-NOV-16	11166	27
28	1000000058	16-NOV-16	12941	28
29	1000000059	11-NOV-16	14241	29
30	1000000060	06-APR-16	9391	30

PROCEDURES:

/*1 A procedure to calculate the total income of all customers: */

This SQL script defines a stored procedure named total_customer_income to calculate the sum of incomes for all customers stored in the CUSTOMER table. After execution, it outputs the total income using DBMS_OUTPUT.PUT_LINE.

```
CREATE OR REPLACE PROCEDURE total_customer_income AS
   total_income NUMBER;
BEGIN
   SELECT SUM(Income) INTO total_income
   FROM CUSTOMER;

   DBMS_OUTPUT.PUT_LINE('Total income of all customers: ' || total_income);
END;
/
BEGIN
  total_customer_income();
END;
/
```

```
Statement processed.
Total income of all customers: 3852700
```

/*2 A procedure to find the agent with the highest commission: */

This SQL procedure, find_highest_commission, identifies the agent with the highest commission by querying the AGENT table and retrieving the name and commission value where the commission equals the maximum commission in the table

```
CREATE OR REPLACE PROCEDURE find_highest_commission AS
   highest_commission AGENT.Name%TYPE;
   highest_commission_value AGENT.Commission%TYPE;
BEGIN
   SELECT Name, Commission
   INTO highest_commission, highest_commission_value
   FROM AGENT
   WHERE Commission = (SELECT MAX(Commission) FROM AGENT);

   DBMS_OUTPUT.PUT_LINE('Agent with the highest commission: ' || highest_commission || '
- ' || highest_commission_value);
END;
/
BEGIN
  find_highest_commission();
END;
/
```

```
Statement processed.
Agent with the highest commission: Bernardine Tayloe - 34900
```

/*3 A procedure to find the customer with the most policies: */

This SQL procedure, find_most_policies, retrieves the customer with the highest number of policies by counting the policies for each customer in the CUSTOMER table and selecting the customer with the maximum count.

```
CREATE OR REPLACE PROCEDURE find_most_policies AS
   most_policies VARCHAR2(100); -- Adjust the size according to your needs
BEGIN
   SELECT Fname || ' ' || Lname
   INTO most_policies
   FROM (
      SELECT Fname, Lname, COUNT(*) AS num_policies
      FROM CUSTOMER
      GROUP BY Fname, Lname
      ORDER BY COUNT(*) DESC
   )
   WHERE ROWNUM = 1;

   DBMS_OUTPUT.PUT_LINE('Customer with the most policies: ' || most_policies);
END;
/
BEGIN
```

```
  find_most_policies();

END;

/
```

```
Statement processed.
Customer with the most policies: Karin Rothermel
```

/*4 A procedure to find the average coverage of all policies: */

This SQL procedure, average_coverage, calculates the average coverage amount across all policies stored in the POLICY table. It computes the average coverage by querying the POLICY table and using the AVG() function.

```
CREATE OR REPLACE PROCEDURE average_coverage AS

    avg_coverage NUMBER; -- Declare the variable



BEGIN

    SELECT AVG(Coverage) INTO avg_coverage

    FROM POLICY;



    DBMS_OUTPUT.PUT_LINE('Average coverage of all policies: ' || avg_coverage);

END;

/

BEGIN

  average_coverage();

END;

/
```

```
Statement processed.
Average coverage of all policies: 23500000
```

CURSORS AND TRIGGERS:

/*1 CURSORS*/

This PL/SQL block demonstrates the usage of cursors to iterate through records in the CUSTOMER table. It declares a cursor named c_customer to fetch all columns from the CUSTOMER table. Inside the loop, it fetches each row into the c_customer_row variable and prints the customer's name

```
DECLARE
  CURSOR c_customer IS
    SELECT * FROM CUSTOMER;
  c_customer_row CUSTOMER%ROWTYPE; -- Declare c_customer_row using the %ROWTYPE attribute
BEGIN
  OPEN c_customer;
  LOOP
    FETCH c_customer INTO c_customer_row;
    EXIT WHEN c_customer%NOTFOUND;
    -- Perform some actions with the fetched row
    DBMS_OUTPUT.PUT_LINE('Customer Name: ' || c_customer_row.Fname || ' ' || c_customer_row.Lname);
  END LOOP;
  CLOSE c_customer;
```

```
END;

/
```

```
Statement processed.
Customer Name: Shaine Plastow
Customer Name: Phillida Farland
Customer Name: Maximilian Deniskevic
Customer Name: Rosalinda Bloore
Customer Name: Gloriana Gebhardt
Customer Name: Pierrette Kilmary
Customer Name: Melisande Lyptratt
Customer Name: Marty Dalgliesh
Customer Name: Karin Rothermel
Customer Name: Emmott Dominicacci
Customer Name: Jarvis Muehler
Customer Name: Pamela Castellani
Customer Name: Orson Drewe
Customer Name: Marabel MacRirie
Customer Name: Bobine Purver
Customer Name: Edouard Thompson
Customer Name: Feliza Kaplan
Customer Name: Regina Alleyne
Customer Name: Winslow Argente
Customer Name: Moises McGahern
Customer Name: Leah Laterza
Customer Name: Alanna Slater
Customer Name: Judah Bruffell
Customer Name: Rancell Rabb
Customer Name: Isac Yacob
Customer Name: Addi McInnerny
Customer Name: Amil Solly
Customer Name: Orton Dimitrov
Customer Name: Ugo Toffanini
Customer Name: Oswell Sango
```

/*2 TRIGGERS*/

This trigger, named "agent_default_policy_count," is designed to execute before an insertion operation on the AGENT table. It ensures that if the column "Num_of_policy_done" is not specified during insertion, it defaults to 0 for the new agent.

```
CREATE OR REPLACE TRIGGER agent_default_policy_count

BEFORE INSERT ON AGENT

FOR EACH ROW

BEGIN

   IF :NEW.Num_of_policy_done IS NULL THEN

      :NEW.Num_of_policy_done := 0; -- Set default value to 0

   END IF;

END;

/

-- Insert a new agent without specifying Num_of_policy_done

INSERT INTO AGENT (Name, Ssn, Salary, Commission) VALUES ('John Doe', 1000000031, 80000, 5000);

-- Query the AGENT table to see the updated values

SELECT * FROM AGENT WHERE Ssn = 1000000031;
```

NAME	SSN	SALARY	COMMISSION	NUM_OF_POLICY_DONE
John Doe	1000000031	80000	5000	0

FUNCTIONS:

/*1 To select the name and address of the agents who work in Florida*/

This SQL script defines a function named "agent_florida_names" to retrieve the names of agents working in Florida along with their addresses. It utilizes a subquery to filter agents based on their addresses in Florida. The function aggregates the names using LISTAGG and returns them. The subsequent execution of the function retrieves the agent names in Florida

```
CREATE OR REPLACE FUNCTION agent_florida_names

RETURN VARCHAR2

IS

  agent_names VARCHAR2(500);

BEGIN

  SELECT LISTAGG(name, ', ') WITHIN GROUP (ORDER BY name) INTO agent_names

  FROM agent

  WHERE ssn IN (SELECT ssn FROM address WHERE state = 'Florida');



  RETURN agent_names;

END;

/-- Now let's call the function

DECLARE

  agent_list VARCHAR2(500);

BEGIN
```

```
  agent_list := agent_florida_names();

  DBMS_OUTPUT.PUT_LINE('Agents in Florida: ' || agent_list);

END;

/
```

```
Statement processed.
Agents in Florida: Berrie Chaudrelle, Curtis Clausen-Thue, Elyse Wells, Obidiah Heathorn, Peterus Derwin, Zak Fleetham
```

/*2 To select ssn first name and last name of customers with active policies*/

This SQL script defines a function named "customer_active_policy" to fetch the Social Security Number (SSN), first name, and last name of customers with active policies. The function uses a cursor to query the customer and cust_policy tables, joining them on the policy number to retrieve the required information. Upon execution, the function returns a cursor containing the queried data. The subsequent block retrieves and prints SSN, first name, and last name of customers with active policies using the returned cursor. This approach efficiently retrieves and processes data within the database.

```
CREATE OR REPLACE FUNCTION customer_active_policy

RETURN SYS_REFCURSOR

IS

  customer_policies SYS_REFCURSOR;

BEGIN

  OPEN customer_policies FOR

    SELECT c.ssn, c.fname, c.lname

    FROM customer c

    JOIN cust_policy cp ON c.policy_num = cp.policy_num;

  RETURN customer_policies;

END;

/

DECLARE

  customer_policy_cursor SYS_REFCURSOR;
```

```
  v_ssn customer.ssn%TYPE;
  v_fname customer.fname%TYPE;
  v_lname customer.lname%TYPE;
BEGIN
  customer_policy_cursor := customer_active_policy();
  LOOP
    FETCH customer_policy_cursor INTO v_ssn, v_fname, v_lname;
    EXIT WHEN customer_policy_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('SSN: ' || v_ssn || ', Name: ' || v_fname || ' ' || v_lname);
  END LOOP;
  CLOSE customer_policy_cursor;
END;
/
```

```
Statement processed.
SSN: 1000000031, Name: Shaine Plastow
SSN: 1000000032, Name: Phillida Farland
SSN: 1000000033, Name: Maximilian Deniskevich
SSN: 1000000034, Name: Rosalinda Bloore
SSN: 1000000035, Name: Gloriana Gebhardt
SSN: 1000000036, Name: Pierrette Kilmary
SSN: 1000000037, Name: Melisande Lyptratt
SSN: 1000000038, Name: Marty Dalgliesh
SSN: 1000000039, Name: Karin Rothermel
SSN: 1000000040, Name: Emmott Dominicacci
SSN: 1000000041, Name: Jarvis Muehler
SSN: 1000000042, Name: Pamela Castellani
SSN: 1000000043, Name: Orson Drewe
SSN: 1000000044, Name: Marabel MacRirie
SSN: 1000000045, Name: Bobine Purver
SSN: 1000000046, Name: Edouard Thompson
SSN: 1000000047, Name: Feliza Kaplan
SSN: 1000000048, Name: Regina Alleyne
SSN: 1000000049, Name: Winslow Argente
SSN: 1000000050, Name: Moises McGahern
SSN: 1000000051, Name: Leah Laterza
SSN: 1000000052, Name: Alanna Slater
SSN: 1000000053, Name: Judah Bruffell
SSN: 1000000054, Name: Rancell Rabb
SSN: 1000000055, Name: Isac Yacob
SSN: 1000000056, Name: Addi McInnerny
SSN: 1000000057, Name: Amil Solly
SSN: 1000000058, Name: Orton Dimitrov
SSN: 1000000059, Name: Ugo Toffanini
SSN: 1000000060, Name: Oswell Sango
```

/*3 To select ssn first name and last name of customers who have claimed their policies */

This SQL script defines a function named "customer_claimed_policy" to retrieve the Social Security Number (SSN), first name, and last name of customers who have claimed their policies. It iterates through customers with claims, concatenates their information into a string, and returns it. Upon execution, the function returns a string containing the SSN, first name, and last name of customers who have claimed policies.

```
CREATE OR REPLACE FUNCTION customer_claimed_policy

RETURN VARCHAR2

IS

  customer_names VARCHAR2(4000) := '';

BEGIN

  FOR c IN (SELECT ssn, fname, lname FROM customer WHERE ssn IN (SELECT ssn FROM claim))

  LOOP

    customer_names := customer_names || c.ssn || ' ' || c.fname || ' ' || c.lname || ', ';

  END LOOP;

  -- Remove the last comma and space

  IF LENGTH(customer_names) > 2 THEN

    customer_names := SUBSTR(customer_names, 1, LENGTH(customer_names) - 2);

  END IF;

  RETURN customer_names;
```

```
END;
/
DECLARE
  result VARCHAR2(4000); -- Increase the size to accommodate larger strings
BEGIN
  result := customer_claimed_policy();
  DBMS_OUTPUT.PUT_LINE('Customers who have claimed a policy: ' || result);
END;
/
```

```
Statement processed.
Customers who have claimed a policy: 1000000031 Shaine Plastow, 1000000032 Phillida Farland, 1000000033 Maximilian Deniskevich
1000000034 Rosalinda Bloore, 1000000035 Gloriana Gebhardt, 1000000036 Pierrette Kilmary, 1000000037 Melisande Lyptratt
:, 1000000038 Marty Dalgliesh, 1000000039 Karin Rothermel, 1000000040 Emmott Dominicacci, 1000000041 Jarvis Muehler
, 1000000042 Pamela Castellani, 1000000043 Orson Drewe, 1000000044 Marabel MacRirie, 1000000045 Bobine Purver, 1000000046 Edouard Thompson,
1000000047 Feliza Kaplan, 1000000048 Regina Alleyne, 1000000049 Winslow Argente, 1000000050 Moises McGahern, 1000000051 Leah Laterza,
1000000052 Alanna Slater, 1000000053 Judah Bruffell, 1000000054 Rancell Rabb, 1000000055 Isac Yacob, 1000000056 Addi McInnerny,
1000000057 Amil Solly, 1000000058 Orton Dimitrov, 1000000059 Ugo Toffanini, 1000000060 Oswell Sango, 123456789  John Doe
```

QUERIES

/*1) To select ssn first name and last name of customers who have made payments more than once*/

This SQL query retrieves the Social Security Number (SSN), first name, last name, and policy number of customers who have made payments more than once. It performs a join between the customer and payment tables on the SSN attribute, filtering records where the payment number is greater than 2 to identify customers with multiple payments.

select customer.ssn,fname,lname,customer.policy_num from customer, payment where customer.ssn = payment.ssn and payment_no > 2;

SSN	FNAME	LNAME	POLICY_NUM
1000000033	Maximilian	Deniskevich	6
1000000034	Rosalinda	Bloore	12
1000000035	Gloriana	Gebhardt	14
1000000036	Pierrette	Kilmary	18
1000000037	Melisande	Lyptratt	13
1000000038	Marty	Dalgliesh	21
1000000039	Karin	Rothermel	15
1000000040	Emmott	Dominicacci	22
1000000041	Jarvis	Muehler	27
1000000042	Pamela	Castellani	10
1000000043	Orson	Drewe	23
1000000044	Marabel	MacRirie	4
1000000045	Bobine	Purver	1
1000000046	Edouard	Thompson	16
1000000047	Feliza	Kaplan	2
1000000048	Regina	Alleyne	3
1000000049	Winslow	Argente	8
1000000050	Moises	McGahern	17
1000000051	Leah	Laterza	19
1000000052	Alanna	Slater	20
1000000053	Judah	Bruffell	24
1000000054	Rancell	Rabb	25
1000000055	Isac	Yacob	7
1000000056	Addi	McInnerny	26
1000000057	Amil	Solly	28
1000000058	Orton	Dimitrov	29

/*2) To select name and password of the agents*/

This SQL query retrieves the name and password of agents by joining the agent and agnt_password tables based on the SSN attribute. By correlating the SSN values between the two tables, it ensures that the agent's name and corresponding password are selected accurately.

select name, password from agent, agnt_password where agent.ssn = agnt_password.ssn;

NAME	PASSWORD
Obidiah Heathorn	YoJqDBn6L
Ricca Artinstall	Y2k7u2LfcdFu
Bernardine Tayloe	cyr6XR8qFB
Murray Philippou	N4uoBSL0UQ
Berrie Chaudrelle	YW494YQ
Minetta Hodjetts	AW05O1xAc4F3
Marylee Kinnach	RBpAUoCFWDzh
Curtis Clausen-Thue	kqcj3ZCacu
Peterus Derwin	veu4cx4EotE
Farlay Varfalameev	LI3CmPoWXYN3
Harmony Ewles	CWhR7FCr
Simonette Odby	LpHxnYnY
Lucille Skein	UEA8y24aPvV
Katrina Leaburn	VNyPTwfNvSs
Othilie Eggleson	idLenTR342
Elyse Wells	evO6x1doy
Zak Fleetham	nYNv0tL4gAZ
Tine Robert	xUdRifCNlT
Tatiania Dashwood	kJ45l7ztNr
Kalli Mould	XgwxAT
Stacia Egalton	GPPUDfhno
Hollis Persich	Z7BIDCX
Bonny Janicijevic	HCun2at88HeF
Kane Grcic	d6ddJwMoSX
Etan Bernardez	d6ddJklpd
Darb Spriggin	sSkH3CgQ
See Musso	7Na6GF
Michelina Plank	AJTT2b
Domenic Wooffitt	y7kf2NxXqiB
Standford Scarre	mNbo0Q1F

/*3) To select names of agent who work in texas*/

This SQL query retrieves the names of agents who work in Texas by selecting records from the agent table where the SSN matches those stored in the address table with the state attribute set to 'Texas'. By using a subquery, it filters agents based on their corresponding addresses, ensuring accurate selection of agents working in Texas.

select name from agent where ssn in (select ssn from address where state='Texas');

NAME
Bernardine Tayloe
Farlay Varfalameev
Stacia Egalton

/*4) To select maximum salary*/

This SQL query retrieves the maximum salary among all agents stored in the agent table. By utilizing the MAX() function, it efficiently identifies the highest salary value present in the salary column. This query is useful for quickly determining the maximum salary within the agent dataset, aiding in salary analysis or comparison.

select max(salary) from agent;

MAX(SALARY)
178000

/*5) To select the second highest salary*/

This SQL query retrieves the second highest salary from the agent table by excluding the maximum salary and then selecting the maximum from the remaining salaries. By filtering out the highest salary using a subquery, it effectively identifies the second highest salary value.

select max(salary)from agent where salary not in (select max(salary) from agent);

MAX(SALARY)
171300

/*6) To select first name and last name of the customers who have agent associated with them*/

This SQL query selects the first name and last name of customers who have an associated agent by joining the customer and agent tables on the agent_id and SSN attributes respectively. It correlates customer-agent relationships using the agent_id and SSN values, ensuring accurate selection of customer details.

select c.fname,c.lname from (customer as c join (agent as a))where c.agent_id = a.ssn;

fname	lname
Shaine	Plastow
Phillida	Farland
Maximilian	Deniskevich
Rosalinda	Bloore
Gloriana	Gebhardt
Pierrette	Kilmary
Melisande	Lyptratt
Marty	Dalgliesh
Karin	Rothermel
Emmott	Dominicacci
Jarvis	Muehler
Pamela	Castellani
Orson	Drewe
Marabel	MacRirie

fname	lname
Bobine	Purver
Edouard	Thompson
Feliza	Kaplan
Regina	Alleyne
Winslow	Argente
Moises	McGahern
Leah	Laterza
Alanna	Slater
Judah	Bruffell
Rancell	Rabb
Isac	Yacob
Addi	McInnerny
Amil	Solly
Orton	Dimitrov

Ugo	Toffanini
Oswell	Sango

/*7) To get the number of people who work in the state of Florida*/

This SQL query retrieves the count of agents who work in the state of Florida by selecting records from the agent table where the SSN matches those stored in the address table with the state attribute set to 'Florida'. By using a subquery to filter agents based on their corresponding addresses, it accurately determines the number of agents working in Florida.

```
select count(name) from agent where SSN in (select SSN from Address where state='Florida');
```

COUNT(NAME)
6

CONCLUSION:

In conclusion, the SQL project has been successfully completed with the implementation of various procedures, functions, triggers, and sql queries. These database objects were designed to ensure data consistency, accuracy, and integrity.

While functions were developed to carry out particular tasks, such as computing values, manipulating data, and returning results, procedures were used to streamline complex operations. In order to maintain consistency and accuracy of the data, trigger was set up to carry out specific actions automatically when specific circumstances are satisfied. In order to deal with potential issues that can occur while running SQL code, exception handling was implemented. Few important queries to know SQL were also implemented.

Overall, the project has achieved its objectives of creating a robust and reliable database system. The various database objects implemented have helped to streamline data management and improve data quality. With continued maintenance and updates, this SQL project will continue to provide a stable and efficient data platform for the organization.

REFERENCES:

• https://www.javatpoint.com/pl-sql-tutorial

• https://www.geeksforgeeks.org/plsql-introduction/

• https://www.oracletutorial.com/plsql-tutorial/what-is-plsql/

• https://www.tutorialspoint.com/plsql/index.htm

• https://www.oracle.com/in/database/technologies/appdev/plsql.html